

它不仅是一本Selenium2 自动化测试书  
同时还是一本实用的Python基础编程书

Broadview®  
www.broadview.com.cn



# Selenium 2

## 自动化测试实战

### 基于Python语言

—— 虫师 编著 ——



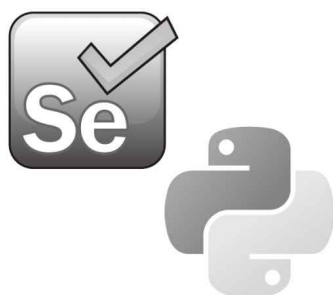
中国工信出版集团



电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
http://www.phei.com.cn

Selenium 2 自动化测试实战 基于Python语言

电子工业出版社



# Selenium 2

## 自动化测试实战

### 基于Python语言

—— 虫师 编著 ——

电子工业出版社  
Publishing House of Electronics Industry  
北京•BEIJING

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

**00001400010000000000000000002000100000000000**

**000000000000000000110000000000000000000000000012**

**0001400**

[illegible]





01088254888

zlbs@phei.com.cn  
dbqq@phei.com.cn

01088258888



[illegible]

# Selenium 2——Python

测试框架  
 测试框架  
 测试框架  
 xunit rspec BDD  
 cucumber UI aut Selenium  
 测试框架

iPad

[illegible][illegible][illegible]

Selenium



GoogleGmailAPI SeleniumAPI

WebDriverAPI

BDDCI



\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

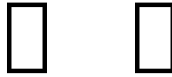
**QUESTION**

1. 在本文中，我们将介绍如何使用Python和BDD（行为驱动开发）来测试我们的应用程序。我们将使用Pytest和Behave库来实现。

2. 首先，我们需要安装Pytest和Behave。可以通过以下命令安装：

3.





---

[目次](#)

[目次](#)

[第1章 環境構築](#)

[1.1 環境構築](#)

[1.2 環境構築](#)

[1.3 環境構築](#)

[1.4 環境構築](#)

[1.5 Selenium環境](#)

[1.6 環境構築](#)

[1.7 環境構築](#)

[1.8 環境構築](#)

[第2章 環境構築](#)

[2.1 Windows環境構築](#)

[2.1.1 Python](#)

[2.1.2 setuptoolsとpip](#)

[2.1.3 Selenium](#)

[2.1.4 ActivePython](#)

[2.2 Ubuntu環境構築](#)

[2.3 IDLEとPython](#)

[2.4 環境構築](#)

[2.5 環境構築](#)

[2.6 環境構築とWebDriver](#)

[第3章 Python](#)

### 3.1 Python

### 3.2

#### 3.2.1 print

#### 3.2.2 input

#### 3.2.3

### 3.3

#### 3.3.1 if

#### 3.3.2 for

### 3.4

#### 3.4.1

#### 3.4.2

### 3.5

#### 3.5.1

#### 3.5.2

### 3.6

#### 3.6.1

#### 3.6.2

#### 3.6.3

#### 3.6.4

### 3.7

#### 3.7.1

#### 3.7.2

#### 3.7.3

### 

## 4 WebDriver API

### 4.1

#### 4.1.1 id

[4.1.2 name](#)

[4.1.3 class](#)

[4.1.4 tag](#)

[4.1.5 link](#)

[4.1.6 partial link](#)

[4.1.8 CSS](#)

[4.1.9 By](#)

[4.2](#)

[4.2.1](#)

[4.2.2](#)

[4.2.3](#)

[4.3](#)

[4.3.1 126](#)

[4.3.2 WebElement](#)

[4.4](#)

[4.5](#)

[4.6](#)

[4.7](#)

[4.7.1](#)

[4.7.2](#)

[4.7.3 sleep](#)

[4.8](#)

[4.9](#)

[4.10](#)

[4.11](#)

[4.12](#)

[4.12.1 send\\_keys](#)

#### 4.12.2 Autolt

#### 4.13

#### 4.14 Cookie

#### 4.15 JavaScript

#### 4.16 HTML5

#### 4.17

#### 4.18

#### 4.19

#### 4.20 WebDriver

#### 

#### 5

#### 5.1

#### 5.1.1

#### 5.1.2

#### 5.1.3

#### 5.1.4

#### 5.2

#### 5.3

#### 5.3.1

#### 5.3.2

#### 5.3.3 txt

#### 5.3.4 csv

#### 5.3.5 xml

#### 

#### 6 Selenium IDE

#### 6.1 Selenium IDE

#### 6.1.1

### 6.1.2 [unittest](#)

## 6.2 [Selenium IDE](#)

### 6.3 [unittest](#)

#### 6.3.1 [unittest](#)

#### 6.3.2 [unittest](#)

## 6.4 [Selenium IDE](#)

### 6.5 [unittest](#)

#### 6.5.1 [unittest](#)

#### 6.5.2 [unittest](#)

### 6.6 [unittest](#)

#### 6.6.1 [unittest](#)

#### 6.6.2 [unittest](#)

### unittest

## 7 [unittest](#)

### 7.1 [unittest](#)

#### 7.1.1 [unittest](#)

#### 7.1.2 [unittest](#)

#### 7.1.3 [unittest](#)

#### 7.1.4 [unittest](#)

#### 7.1.5 [discover](#)

### 7.2 [unittest](#)

#### 7.2.1 [unittest](#)

#### 7.2.2 [unittest](#)

#### 7.2.3 [unittest](#)

#### 7.2.4 [fixtures](#)

### 7.3 [unittest](#)

### 7.4 [Web](#)

[□□□□](#)

[□8□ □□□□□□□□](#)

[8.1 HTML□□□□](#)

[8.1.1 □□HTMLTestRunner](#)

[8.1.2 □□HTML□□□□](#)

[8.1.3 □□□□□□□□](#)

[8.1.4 □□□□□□□□](#)

[8.1.5 □□□□□□□□](#)

[8.2 □□□□□□□□](#)

[8.2.1 □□HTML□□□□□□](#)

[8.2.2 □□□□□□□□](#)

[8.2.3 □□□□□□□□□□](#)

[8.2.4 □□□□□□□□□□](#)

[8.3 Page Object□□□□](#)

[8.3.1 □□Page Object](#)

[8.3.2 Page Object□□](#)

[□□□□](#)

[□9□ Selenium Grid2](#)

[9.1 Selenium Server□□□□](#)

[9.2 Selenium Grid□□□□](#)

[9.3 Remote□□](#)

[9.3.1 WebDriver□□□□](#)

[9.3.2 Remote□□](#)

[9.3.3 □□□□□□□□□□](#)

[9.4 WebDriver□□](#)

[9.4.1 Edge□□□](#)

[9.4.2 Opera□□□](#)

[9.4.3 Safari](#)

[9.4.4 HtmlUnit](#)

[9.4.5 PhantomJS](#)

[10](#)

[10 Python](#)

[10.1](#)

[10.2](#)

[10.2.1 threading](#)

[10.2.2](#)

[10.2.3](#)

[10.3](#)

[10.3.1 multiprocessing](#)

[10.3.2 PipeQueue](#)

[10.4](#)

[10.4.1](#)

[10.4.2](#)

[11](#)

[11](#)

[11.1](#)

[11.1.1](#)

[11.1.2](#)

[11.1.3](#)

[11.2 BBS](#)

[11.2.1](#)

[11.2.2](#)

[11.2.3](#)

[11.2.4 Page Object](#)

[11.2.5 测试用例](#)

[11.2.6 测试用例](#)

[测试用例](#)

[12 BDD测试Lettuce](#)

[12.1 测试BDD](#)

[12.2 测试Lettuce](#)

[12.3 测试用例](#)

[12.3.1 测试用例](#)

[12.3.2 测试BDD](#)

[12.3.3 测试用例](#)

[12.3.4 Lettuce测试用例](#)

[12.4 Lettuce\\_webdriver测试用例](#)

[测试用例](#)

1314测试用例测试用例

[13 GitHub测试用例](#)

[13.1 测试用例](#)

[13.1.1 测试GitHub](#)

[13.1.2 测试Git](#)

[13.1.3 测试用例](#)

[13.2 Git/GitHub测试用例](#)

[13.2.1 GitHub测试用例](#)

[13.2.2 测试用例](#)

[13.2.3 测试用例](#)

[13.2.4 测试用例](#)

[测试用例](#)

[14 测试Jenkins](#)

[14.1 测试用例](#)



14.2 □□□□

14.3 □□□□

14.4 □□□□□□

□□□□



# 1 第一章

本章主要介绍本章的主要内容，包括本章的学习目标和重点。

## 1.1 第一节

本节主要介绍第一节的主要内容，包括本节的学习目标和重点。

### 1.1.1 1.1.1

1.1.1 本节主要介绍1.1.1节的主要内容，包括本节的学习目标和重点。

1.1.1.1 本节主要介绍1.1.1.1节的主要内容，包括本节的学习目标和重点。

2.1.1 本节主要介绍2.1.1节的主要内容，包括本节的学习目标和重点。

3.1.1 本节主要介绍3.1.1节的主要内容，包括本节的学习目标和重点。

4.1.1 本节主要介绍4.1.1节的主要内容，包括本节的学习目标和重点。

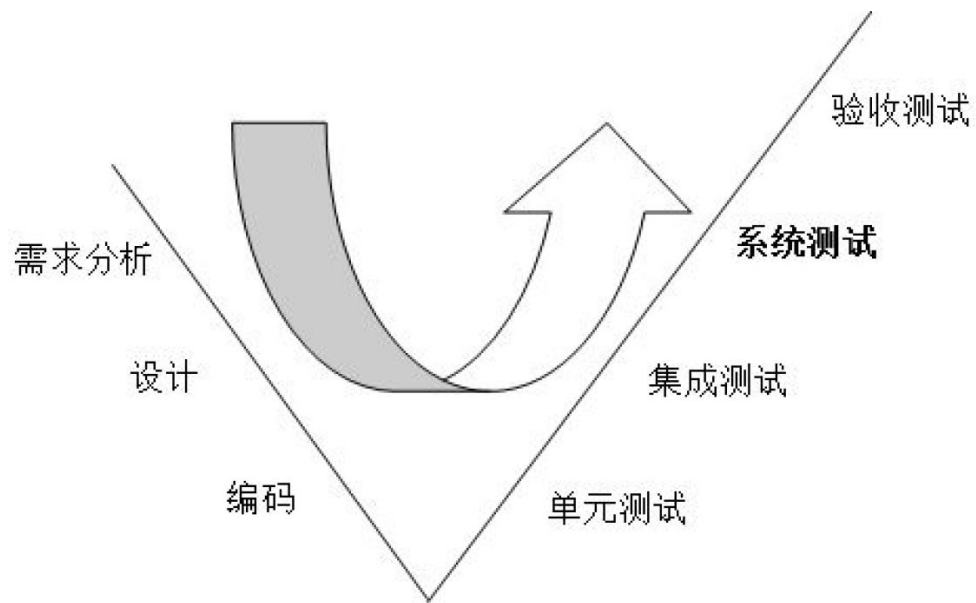
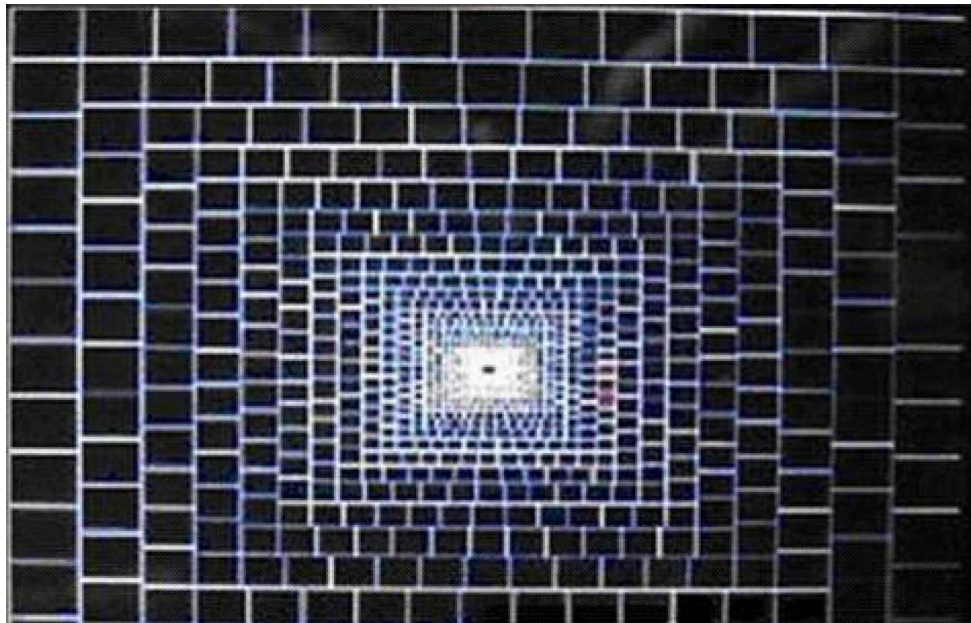


图1.1 软件开发生命周期

## 2. 软件测试的分类

软件测试可以分为静态测试和动态测试。静态测试包括需求测试、设计测试和代码测试。动态测试包括单元测试、集成测试、系统测试和验收测试。

图1.2 软件测试的分类



## 1.2 实验目的

1□□□□

[illegible]

## 2□□□□

[illegible][illegible]

## 3□□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

[illegible]

**3**

□ □

## 1. 目的

本報告は、  
の調査結果をまとめたものである。

調査の結果、

## 2. 調査方法

調査は、  
の調査結果をまとめたものである。

調査の結果、

- 調査の結果、  
の調査結果をまとめたものである。  
の調査結果をまとめたものである。  
の調査結果をまとめたものである。
- 調査の結果、CPUの調査結果をまとめたものである。

## 4. 結論

調査の結果、

### 1. 目的

本報告は、  
の調査結果をまとめたものである。







这篇文章来自Mike Cohn的*Succeeding with Agile* 第1.3章，主要讨论了测试策略。文章指出，在敏捷开发中，测试应该是一个持续的过程，而不是一个独立的阶段。作者建议采用“测试驱动开发”（TDD）的方法，即在编写代码之前先编写测试用例。这样可以确保代码的质量和可测试性。文章还提到，测试应该覆盖所有的功能，而不仅仅是单元测试。作者认为，测试应该是一个团队的责任，而不是测试人员的专属工作。最后，文章强调，测试应该是一个持续的过程，而不是一个一次性的任务。

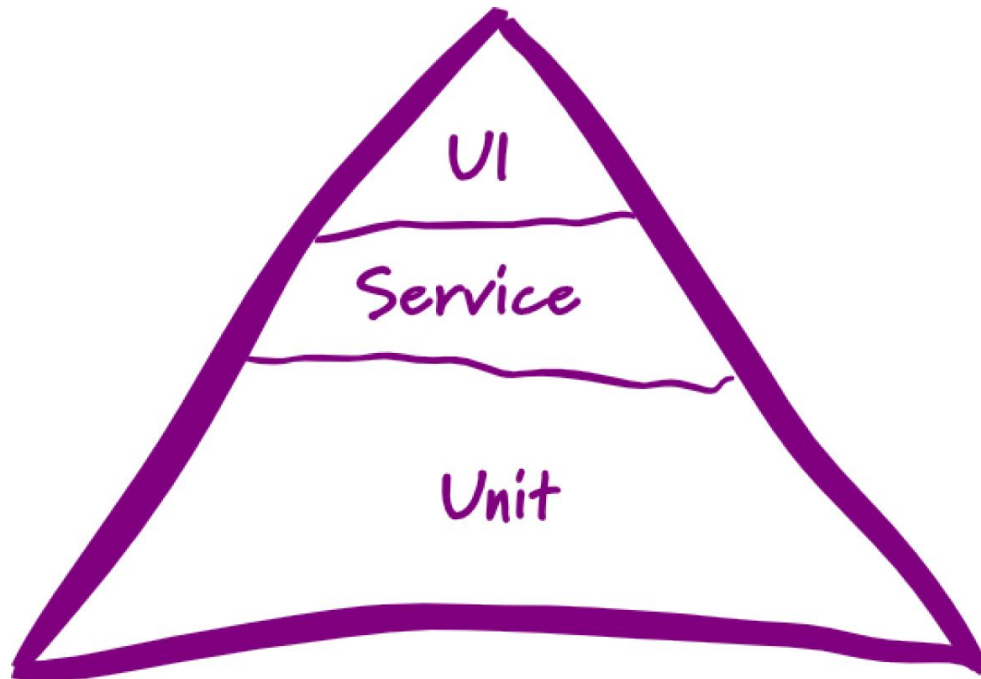


图1.3 测试金字塔

Martin Fowler在《Patterns of Enterprise Application Architecture》中提出了“测试金字塔”（Testing Pyramid）的概念。他认为，测试应该是一个持续的过程，而不是一个独立的阶段。作者建议采用“测试驱动开发”（TDD）的方法，即在编写代码之前先编写测试用例。这样可以确保代码的质量和可测试性。文章还提到，测试应该覆盖所有的功能，而不仅仅是单元测试。作者认为，测试应该是一个团队的责任，而不是测试人员的专属工作。最后，文章强调，测试应该是一个持续的过程，而不是一个一次性的任务。

这篇文章主要讨论了测试策略。文章指出，在敏捷开发中，测试应该是一个持续的过程，而不是一个独立的阶段。作者建议采用“测试驱动开发”（TDD）的方法，即在编写代码之前先编写测试用例。这样可以确保代码的质量和可测试性。文章还提到，测试应该覆盖所有的功能，而不仅仅是单元测试。作者认为，测试应该是一个团队的责任，而不是测试人员的专属工作。最后，文章强调，测试应该是一个持续的过程，而不是一个一次性的任务。

这篇文章主要讨论了测试策略。文章指出，在敏捷开发中，测试应该是一个持续的过程，而不是一个独立的阶段。作者建议采用“测试驱动开发”（TDD）的方法，即在编写代码之前先编写测试用例。这样可以确保代码的质量和可测试性。文章还提到，测试应该覆盖所有的功能，而不仅仅是单元测试。作者认为，测试应该是一个团队的责任，而不是测试人员的专属工作。最后，文章强调，测试应该是一个持续的过程，而不是一个一次性的任务。



## 2. Web 開發

Web 開發是指開發 Web 應用程式的過程。

1. 了解需求與分析：在開發之前，需要與客戶溝通，了解他們的需求，並分析這些需求，以確定開發的範圍和目標。

2. Web 開發技術：

- 了解 Web 開發技術：Web 開發技術包括 HTML/CSS/JavaScript 以及 PHP/Java/C#/Python/Ruby 等。此外，還需要了解 HTTP 協議。
- 了解 Web 開發工具：Web 開發工具包括 IDE、測試工具、部署工具等。

了解 HTTP 協議、HttpUnit、Postman 等。

## 3. UI 測試

UI 測試是指對應用程式的用戶界面進行測試，以確保其正確性和穩定性。常用的 UI 測試工具包括 UFT、Watir、Robot Framework、Selenium 等。

UI 測試技術包括 JavaScript、jQuery、QUnit 等。此外，還需要了解 JavaScript 的測試框架。

[illegible]

UI

UI

UI

“ ”

Google  
Google 70%/20%  
Unit Service UI

[illegible]

### 1.3

[illegible]

1□□□□□□□□□□□□□□



[illegible]

3□□□□□□□□□□

C/S  
B/S

## 1.4

[illegible]

“ ” “ UI ”

[illegible]

1UFT

[illegible]

## 2 Robot Framework

Robot Framework 是一个 Python 实现的开源测试框架，用于自动化测试。它支持多种编程语言，如 Python、Java、C# 等。Robot Framework 的测试用例是用自然语言编写的，易于阅读和维护。

## 3 Watir

Watir (Web Application Testing in Ruby) 是一个 Ruby 实现的开源测试框架，用于自动化测试。它支持多种编程语言，如 Ruby、Python、Java 等。Watir 的测试用例是用 Ruby 编写的，易于阅读和维护。

## 4 Selenium

Selenium 是一个开源的自动化测试工具，用于测试 Web 应用程序。它支持多种编程语言，如 Python、Java、C# 等。Selenium 的测试用例是用代码编写的，易于阅读和维护。

它是一个跨平台的工具，可以在 Windows、Mac OS X 和 Linux 上运行。Selenium 的测试用例是用代码编写的，易于阅读和维护。

# 1.5 Selenium

## 1 Selenium

Selenium 是一个开源的自动化测试工具，用于测试 Web 应用程序。它支持多种编程语言，如 Python、Java、C# 等。Selenium 的测试用例是用代码编写的，易于阅读和维护。

Selenium 支持多种浏览器，包括：

- Firefox
- Chrome
- IE
- Opera
- Edge

- 支持Linux、Windows、MAC
- 支持Java、Python、Ruby、C#、JavaScript、C++
- Web浏览器
- API接口
- 支持API接口

Selenium 1.0 Selenium 2.0 Selenium 1.0 Selenium 1.5

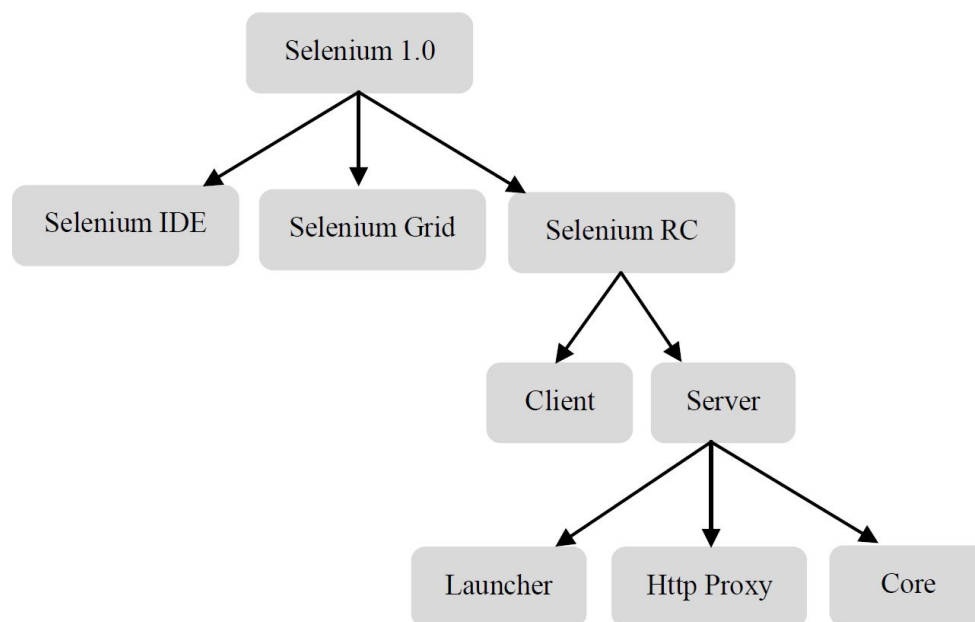


图 1.5 Selenium 1.0 架构

## 2 Selenium IDE

Selenium IDE 支持 Firefox 浏览器，支持 Selenium 1.0 和 Selenium 2.0 版本。



bug IDE bug

IDE

### 3 Selenium Grid

Selenium Grid Grid Web-App Grid

### 4 Selenium RC

Selenium RC Remote Control Selenium Selenium RC Selenium RC

Selenium RC Client Libraries Selenium Server Client Libraries Selenium Server Selenium Server Selenium Server Launcher Http Proxy Core Selenium Core Selenium Server Selenium Core JavaScript JavaScript Launcher Selenium Core Selenium Server Http Proxy

## 5 Selenium 2.0

Selenium 1.0 Selenium 2.0 Selenium 2.0  
 WebDriver

Selenium 2.0 = Selenium 1.0 + WebDriver

Selenium 2.0 WebDriver Selenium  
 RC Selenium Selenium 2.0  
 Selenium RC Selenium  
 WebDriver Selenium RC WebDriver

Selenium RC JavaScript JavaScript  
 selenese selenese Selenium

WebDriver WebDriver  
 Web JavaScript  
 JavaScript WebDriver

Selenium WebDriver WebDriver Simon  
 Stewart 2009 8

**Selenium WebDriver** WebDriver Selenium JavaScript API Selenium WebDriver Selenium

---

## 1.6 Selenium Web 浏览器自动化 Web 浏览器

Selenium Web 浏览器自动化 Web 浏览器  
Web 浏览器

### 1 HTML

HTML Hyper Text Markup Language 超文本标记语言  
超文本标记语言 HTML 超文本标记语言  
VBScript JavaScript HTML

html

```
<html>
  <head>
    <title></title>
  </head>
  <body>
    <h1></h1>
  </body>
</html>
```

<html></html>

<head></head>

```
<title> </title>
```

```
<body> </body>
```

```
<h1> </h1>h1
```

“ ”  
HTML

HTML w3school

## 2 JavaScript

JavaScript Netscape LiveScript  
HTML  
HTML

HTML JavaScript <script> type

js\_page.html

```
<html>
```

```
<body>
```

```
<script type="text/javascript">
```

```
document.write("Hello World!");
```

```
</script>
```

```
</body>
```

```
</html>
```

<script type="text/javascript"></script>
JavaScriptdocument.write()

### 3 XML

XMLHTMLHTML
HTMLXML

XML

xml\_file.xml

<?xml version="1.0"?>

<note>

<to>George</to>

<from>John</from>

<heading>Reminder</heading>

<body>Don't forget the meeting!</body>

</note>

<?xml version="1.0"?>XMLXML

<note></note>

<to></to> <from></from> <heading></heading>
<body></body>XML

XML 文档的根元素是 <?xml> 声明，它定义了文档的根元素。

XML 文档的根元素是 <?xml> 声明，它定义了文档的根元素。XML 文档的根元素是 <?xml> 声明，它定义了文档的根元素。

XML 文档的根元素是 <?xml> 声明，它定义了文档的根元素。XML 文档的根元素是 <?xml> 声明，它定义了文档的根元素。

## 1.7 本章小结

### 1. FireBug

FireBug 是 Firefox 浏览器的一个插件，它提供了对 HTML、JavaScript、Cookie、XMLHttpRequest、CSS、HTML、Ajax 等技术的调试功能。



图 1.6 FireBug

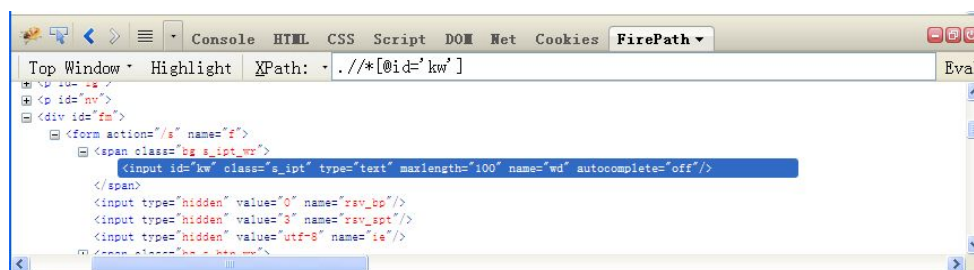
FireBug 是 Firefox 浏览器的一个插件，它提供了对 HTML、JavaScript、Cookie、XMLHttpRequest、CSS、HTML、Ajax 等技术的调试功能。

FireBug 是 Firefox 浏览器的一个插件，它提供了对 HTML、JavaScript、Cookie、XMLHttpRequest、CSS、HTML、Ajax 等技术的调试功能。

FireBug

## 2 FirePath

FirePathFireBugXPath1.0CSS 3jQueryXPathCSS1.7



## □1.7 FirePath

```

    FireBugXPath
    XPath: CSS
    FireBug

```

### 3 Chrome IE

Chrome IE FireBug

Chrome Chrome Chrome  
“ ” → “ ” Ctrl+Shift+I F12  
1.8

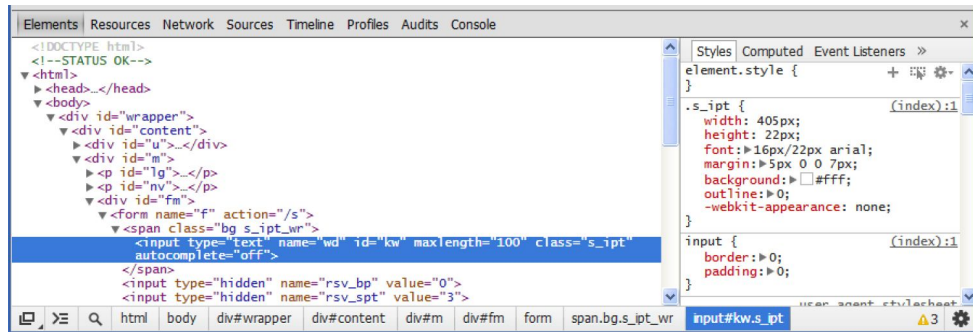


图 1.8 Chrome 元素选择器

IE 浏览器 IE8 浏览器等浏览器中，通过“F12”键打开“F12 开发者工具”，在“F12 开发者工具”中，通过“F12 开发者工具”中的“元素”面板，选择要测试的元素，然后单击“元素”面板中的“元素”按钮，即可打开“元素”面板。

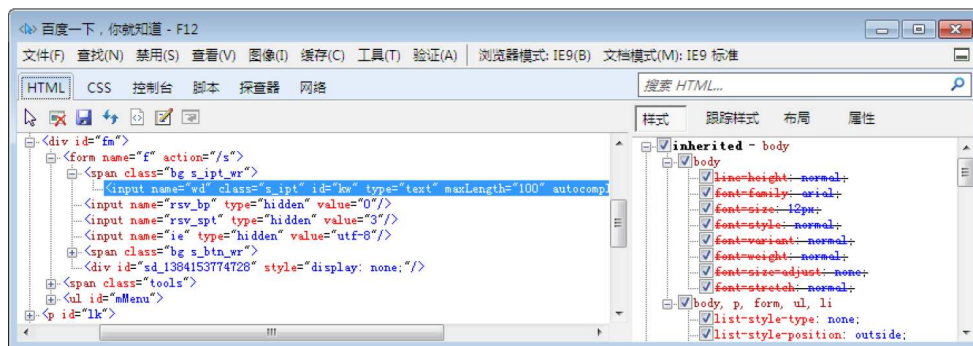


图 1.9 IE 元素选择器

## 1.8 Selenium WebDriver

Selenium WebDriver 是一个跨平台的 Java 库，它允许你使用 Python、Ruby、PHP、C# 或 JavaScript 来编写测试脚本。Selenium WebDriver 是一个跨平台的 Java 库，它允许你使用 Python、Ruby、PHP、C# 或 JavaScript 来编写测试脚本。

Python + Selenium 是一个跨平台的 Java 库，它允许你使用 Python、Ruby、PHP、C# 或 JavaScript 来编写测试脚本。



```

JavaWebPHPWebSelenium

```

[illegible]

☐ Selenium
 ☐ Java
 ☐ C#
 ☐ Python

```

Python Ruby
Python
Ruby Ruby " "
Python

```

Python Web GUI

Python  
Python  
Selenium Python  
Python

`PythonPythonPythonPythonPython`



## 2 環境構築

---

本書では、Python 2 と Python 3 の両方を実行可能な環境を構築します。Python 2 と Python 3 の両方を実行可能な環境を構築するために、Python 2 と Python 3 の両方を実行可能な環境を構築する必要があります。

### 2.1 Windows 環境構築

Windows 環境では、Python 2 と Python 3 の両方を実行可能な環境を構築する必要があります。Python 2 と Python 3 の両方を実行可能な環境を構築するために、Python 2 と Python 3 の両方を実行可能な環境を構築する必要があります。

Python 2 と Python 3 の両方を実行可能な環境を構築するために、Python 2 と Python 3 の両方を実行可能な環境を構築する必要があります。Python 2 と Python 3 の両方を実行可能な環境を構築するために、Python 2 と Python 3 の両方を実行可能な環境を構築する必要があります。

Python 2 と Python 3 の両方を実行可能な環境を構築するために、Python 2 と Python 3 の両方を実行可能な環境を構築する必要があります。Python 2 と Python 3 の両方を実行可能な環境を構築するために、Python 2 と Python 3 の両方を実行可能な環境を構築する必要があります。

Python 2 と Python 3 の両方を実行可能な環境を構築するために、Python 2 と Python 3 の両方を実行可能な環境を構築する必要があります。Python 2 と Python 3 の両方を実行可能な環境を構築するために、Python 2 と Python 3 の両方を実行可能な環境を構築する必要があります。

## 2.1.1 Python

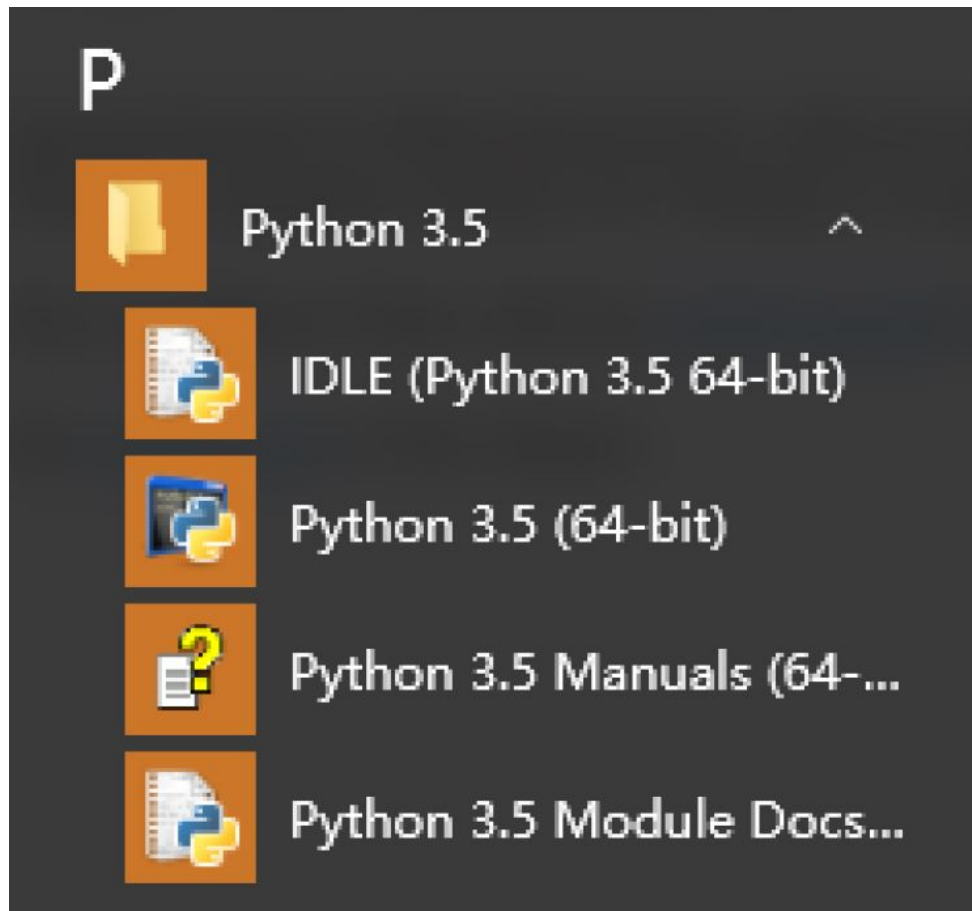
Python <https://www.Python.org/>

Python 3 Python 3.5  
Windows 32 x86  
64 64 .msi 2.1



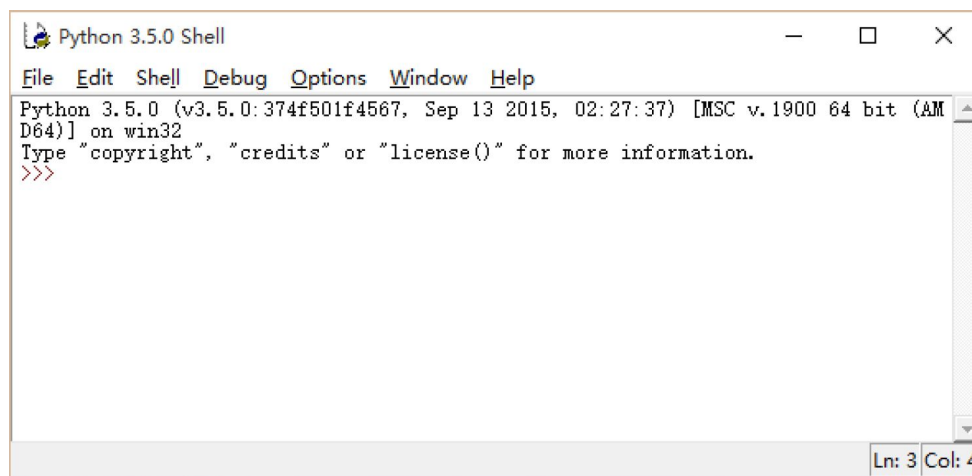
2.1 Python

Windows Python  
2.2



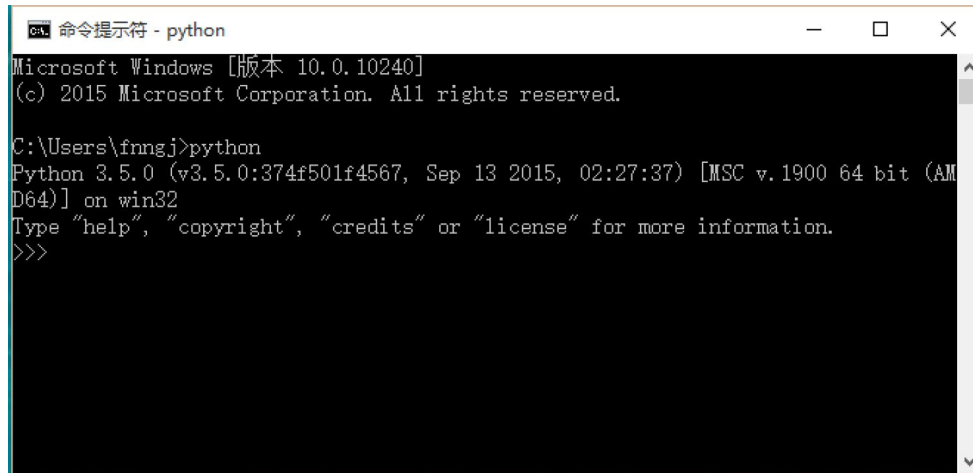
## 2.2 Python

Python IDLE Python Python Shell 2.3



## 2.3 Python Shell

在Windows中通过“python”命令启动Python Shell  
2.4



```
命令提示符 - python
Microsoft Windows [版本 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\fnngj>python
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

## 2.4

在Windows中通过“python”命令启动Python Shell  
Path环境变量“Path”指向Python安装目录  
Path

PATH

;C:\Python35

2.1 Python Add Python 3.5 to PATH  
PATH

## 2.1.2 setuptools pip

```

setuptools Python Enterprise Application Kit PEAK
Python distutilsde Python

```

```

PythonPythonPythonPythonPythonPython
easy_installeasy_installeasy_installeasy_installeasy_installeasy_install
setuptoolssetuptoolssetuptoolssetuptoolssetuptoolssetuptools
easy_installeasy_installeasy_installeasy_installeasy_installeasy_install

```

```

    pip→Python→pip→Python→
    →→→→→pip→setuptools→pip→
    setuptools→Python 3→setuptools→
    distribute
  
```

setuptools pip

<https://pypi.python.org/pypi/setuptools>

<https://pypi.python.org/pypi/pip>

[illegible]

setuptools-18.4.zip

pip-7.1.2.tar.gz

Windows  
python setup.py setuptools pip

cmd.exe

```
C:\package\setuptools-7.0>python setup.py install
```

.....

```
C:\package\pip-1.5.6>python setup.py install
```

```
Python Python pip Python
C:\Python35\Script\pip.exe pip3.exe
Windows pip pip3
```

cmd.exe

```
C:\Users\fnngj>pip
```

Usage:

```
pip<command>[options]
```

Commands:

|                      |                                  |
|----------------------|----------------------------------|
| install              | Install packages.                |
| uninstall            | Uninstall packages.              |
| freeze               | Output installed packages in     |
| requirements format. |                                  |
| list                 | List installed packages.         |
| show                 | Show information about installed |
| packages.            |                                  |
| search               | Search PyPI for packages.        |
| wheel                | Build wheels from your           |
| requirements.        |                                  |
| zip                  | DEPRECATED. Zip individual       |
| packages.            |                                  |





pip 安装 selenium 的步骤如下：  
1. 安装 pip  
2. 安装 Python  
3. 安装 Selenium

cmd.exe

```
C:\Users\fnngj>pip install selenium==2.48.0 //安装 selenium
```

```
C:\Users\fnngj>pip show selenium //查看 selenium 信息
```

---

Metadata-Version:1.1

Name:selenium

Version:2.48.0

Summary:Python bindings for Selenium

Home-page:<https://github.com/SeleniumHQ/selenium/>

Author:UNKNOWN

Author-email:UNKNOWN

License:UNKNOWN

Location:c:\python35\lib\site-packages

Requires:

```
C:\Users\fnngj>pip uninstall selenium //卸载 selenium
```

pip 安装 selenium 的步骤如下：  
1. 安装 pip  
2. 安装 Python  
3. 安装 Selenium

## 2.1.4 ActivePython

ActivePython 是 ActiveState 公司开发的 Python 解释器。

ActivePython 提供了 Python 的集成环境。Python 的集成环境包括 Python 解释器、IDLE 编辑器、Python 的 Windows API 接口等。ActivePython 提供了 Python 的集成环境。

ActivePython 提供了 pip 包管理工具。pip 是 Python 的包管理工具。

ActivePython 提供了

<http://www.activestate.com/activePython/downloads>

ActivePython 提供了 Windows、Mac、Linux 的集成环境。ActivePython 提供了 2.5 版本。



2.5 ActivePython 提供了

ActivePython 2.7.6.0 Python 2.7.6.0 Windows 7  
64-bit

ActivePython 2.7.6.0 pip Selenium

## 2.2 Ubuntu

Linux Ubuntu

Ubuntu Python Ubuntu Python  
Ubuntu Python

Ubuntu Python 2 Python 3  
"python 2" "Python 3" Python Shell

ubuntu

```
fnngj@fnngj-PC:~$python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>>quit()
```

```
fnngj@fnngj-PC:~$python3
Python 3.4.3 (default, Jul 28 2015, 18:20:59)
```

[GCC 4.8.4] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>>quit()

#####Ubuntu#####Python 3#####setuptools#####pip#####  
Ubuntu#####apt-get#####apt-get#####debian#####  
Ubuntu#####Linux#####

#####setuptools#####

ubuntu#####

fnngj@fnngj-PC:~\$sudo apt-get install python3-setuptools

##### apt-get#####root#####apt-get#####  
#####root#####root#####sudo#####sudo#####  
#####root#####

#####sudo apt-get xxxx

#####pip#####

ubuntu#####

fnngj@fnngj-PC:~\$sudo apt-get install python3-pip

Ubuntu apt-get Windows Python  
setup.py

Python 3 pip Selenium

ubuntu

```
fnngj@fnngj-PC:~$python3-m pip install selenium
```

## 2.3 IDLE Python

Integrated Development Environment (IDE) Python IDLE

IDLE Python GUI IDE IDE Python Shell Python

IDLE “” >>> Python Shell 2.6

```

Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("hello python!")
hello python!
>>> 5+6
11
>>> |

```

图 2.6 Python Shell 的截图

在 IDLE 中运行 Python 代码的方法如下：

## 1. 在 Tab 中运行

在 IDLE 中，可以通过以下方式运行 Python 代码：

- 在 IDLE 的 Shell 窗口中，直接输入并运行代码。
- 在 IDLE 的 Editor 窗口中，编写代码后，按下 `F5` 键运行代码。

```

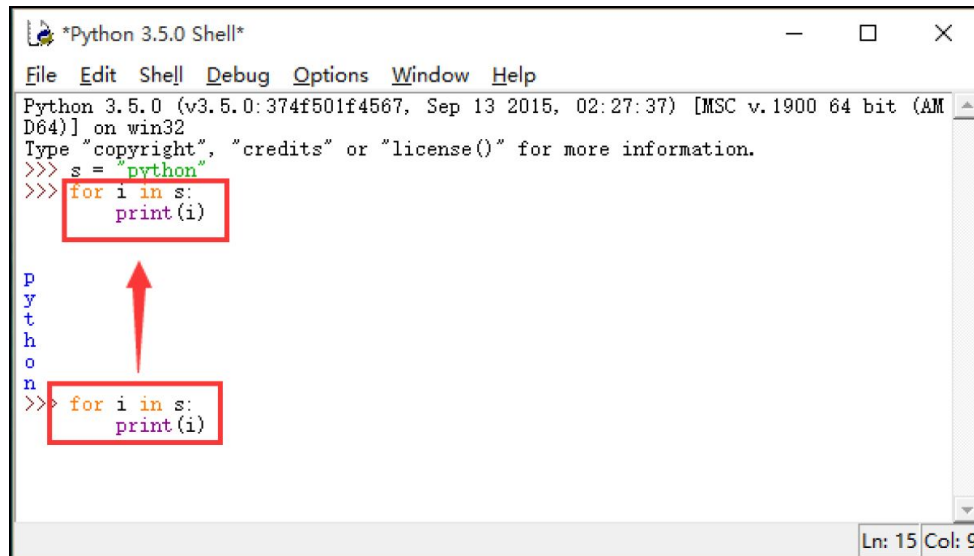
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("hello python!")
hello python!
>>> 5+6
11
>>> pr
print
property
quit
range
repr
reversed
round
set
setattr
slice

```

图 2.7 Tab 的截图

## 2

Alt+P Python Alt+N Python Shell  
Python Shell  
2.8



2.8 Alt+P

Python Shell Python Shell  
File→New File  
Ctrl+N File→Save Ctrl+S  
2.9



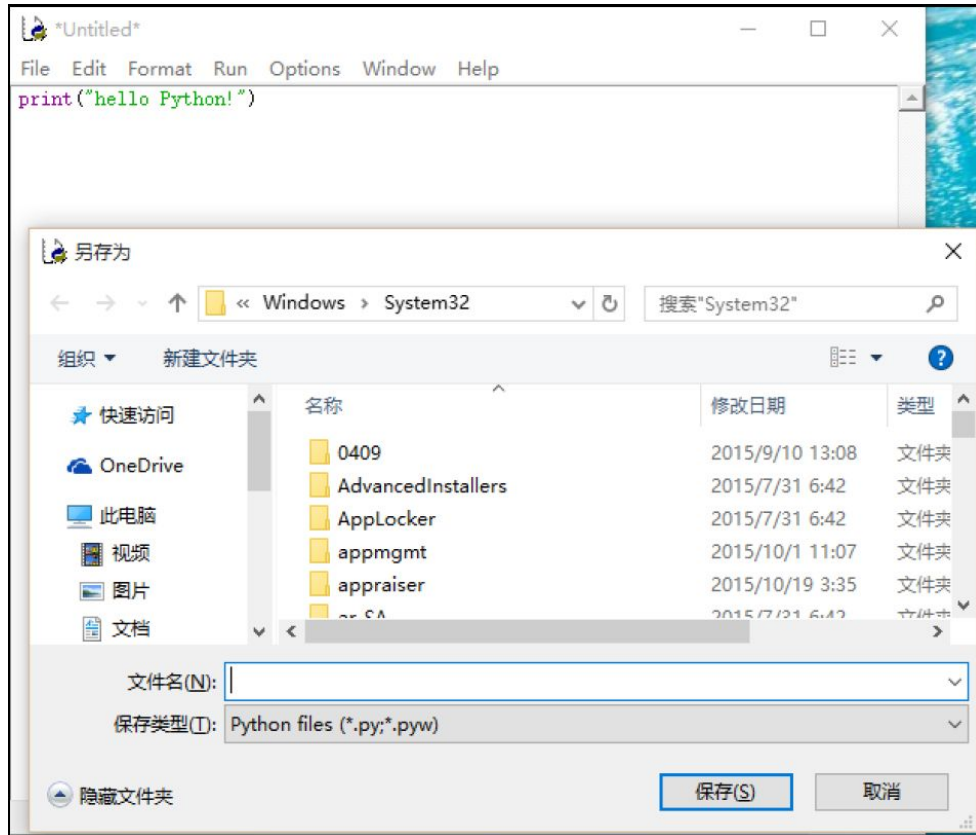


图2.9 保存Python文件

文件扩展名“.py”是Python文件的默认扩展名

## 2.4 安装Python环境

Python IDLE是Python的集成开发环境，它提供了一个简单的Python IDE，可以用来编写和运行Python程序。

baidu.py

```
# coding=utf-8
from selenium import webdriver
```

```
driver=webdriver.Firefox()  
driver.get("http://www.baidu.com")
```

```
driver.find_element_by_id("kw").send_keys("Selenium2")  
driver.find_element_by_id("su").click()  
driver.quit()
```

1. 文件编码问题

```
1 #coding=utf-8
```

2. 文件编码问题

3. 文件编码问题

```
2 #-*-coding:utf-8-*-
```

4. Python 2 和 Python 3 的兼容性

```
3 from selenium import webdriver
```

5. Selenium 和 WebDriver 的区别

WebDriver API 是 Python 的 from... import ... 和 import... 的语法

```
4 driver=webdriver.Firefox()
```

webdriverFirefox driver  
Firefox Selenium WebDriver  
IE Chrome Web

```
5 driver.get("http://www.baidu.com")
```

get() URL

```
6 driver.find_element_by_id("kw").send_keys("Selenium2")
```

id=kw  
send\_keys()“Selenium2”

```
7 driver.find_element_by_id("su").click()
```

id=su“”click()

```
8 driver.quit()
```

baidu.py F5 Firefox  
“Selenium2”  
Firefox 2.10



2.10 环境搭建

## 2.5 环境搭建

WebDriver 支持 Firefox (FirefoxDriver) 支持 IE (InternetExplorerDriver) 支持 Opera (OperaDriver) 支持 Chrome (ChromeDriver) 支持 Android (AndroidDriver) 支持 iPhone (iPhoneDriver) 支持 HtmlUnit 支持 HtmlUnitDriver

环境搭建

<http://www.seleniumhq.org/download/>

Chrome 下载 ChromeDriver\_win32.zip (解压后)  
 解压后 chromedriver.exe 文件位于 Path 路径  
 C:\Python35 路径 Path 路径 chromedriver.exe  
 C:\Python35\

IE 的驱动程序 IEDriverServer\_Win32\_x.xx.zip 解压后  
IEDriverServer.exe 放到 C:\Python35\ 目录下

Linux 的驱动程序 Linux64.tar.gz 解压后放到 Path 目录下  
Linux 的驱动程序 Linux64.tar.gz 解压后放到 Path 目录下

使用 IE、Chrome、Firefox 的驱动程序

```
driver=webdriver.Firefox()
```

```
driver=
```

```
driver=webdriver.Ie()
```

```
driver=
```

```
driver=webdriver.Chrome()
```

使用 IE、Chrome、Firefox 的驱动程序

## 2.6 使用 WebDriver

W3C 的 WebDriver 规范

<http://www.w3.org/TR/webdriver/>

WebDriver 规范使用 WebDriver 规范使用 WebDriver 规范使用  
WebDriver 规范使用 Wire Protocol 规范使用 Web 规范使用

WebDriver API 可以操作 DOM 元素

WebDriver 可以操作 DOM 元素

WebDriver 可以操作 DOM 元素

WebDriver 可以操作 DOM 元素

Java Selenium WebDriver 可以操作 DOM 元素

BaiduTest.java

```
package com.test.case;

//Selenium(webdriver)
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.*;

public class BaiduTest{

    public static void main(String[] args){

        WebDriver driver=new FirefoxDriver();
        driver.get("http://www.baidu.com/");
```

```

        driver.findElement(By.id("kw")).sendKeys("selenium2");
        driver.findElement(By.id("su")).click();

        driver.quit();

    }
}

```

## RubySelenium WebDriver

baidu.rb

```

#Selenium(webdriver)
require 'selenium-webdriver'

driver=Selenium::WebDriver.for:firefox
driver.navigate.to "http://www.baidu.com"

driver.find_element(:id, 'kw').send_keys "selenium2"
driver.find_element(:id, 'su').click()

driver.quit

```

① Seleniumwebdriver

② Seleniumdriver

③ □□□□□□□□URL□

**④**

⑤ □□□□□□□□

```

WebDriver
WebDriver
WebDriver

```







Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one--and preferably only one--obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than \*right\* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea--let's do more of those!

Beautiful is better than ugly□

□□□□□□□

Explicit is better than implicit□

□□□□□□

Simple is better than complex□

□□□□□□

Complex is better than complicated□

□□□□□□

Flat is better than nested□

□□□□□□

Sparse is better than dense□

□□□□□□

Readability counts□

□□□□□□

Special cases aren't special enough to break the rules□

□□□□□□□□□□□□□□□□□□

Although practicality beats purity□

□□□□□□□□

Errors should never pass silently□

□□□□□□□□□□

Unless explicitly silenced

□□□□□□□□

In the face of ambiguity, refuse the temptation to guess

□□□□□□□□□□□□□□□□

There should be one--and preferably only one--obvious way to do it

□□□□□□□□□□□□□□□□□□□□

Although that way may not be obvious at first unless you're Dutch

□□□□□□□□□□□□□□□□Python□□

Now is better than never

□□□□□□□□

Although never is often better than \*right\* now

□□□□□□□□□□

If the implementation is hard to explain, it's a bad idea

□□□□□□□□□□□□□□□□□□□□

If the implementation is easy to explain, it may be a good idea

Namespaces are one honking great idea--let's do more of those

Namespaces are one honking great idea--let's do more of those

Namespaces are one honking great idea--let's do more of those

Namespaces are one honking great idea--let's do more of those

## 3.2 Namespaces

Namespaces are one honking great idea--let's do more of those

### 3.2.1 print

Namespaces are one honking great idea--let's do more of those

Namespaces are one honking great idea--let's do more of those

```
>>>print("hello python")
```

```
hello python
```

Namespaces are one honking great idea--let's do more of those

## Python Shell

```
>>>name="zhangsan"
>>>print("hello%s ,Nice to meet you!"%name)
hello zhangsan ,Nice to meet you!
>>>name="Lisi"
>>>print("hello%s ,Nice to meet you!"%name)
hello Lisi ,Nice to meet you!
```

```
name%sstring%ddata
```

## Python Shell

```
>>>age=27
>>>print("You are%d !"%age)
You are 27 !
```

```
%r
```

## Python Shell

```
>>>n=100
>>>print("You print is%r ."%n)
You print is 100 .
```

```
>>>n="abc"
>>>print("You print is%r ."%n)
You print is 'abc' .
```

```
>>>name="zhangsan"
>>>age=22
>>>print("student info:%s%d ."%(name,age))
student info:zhangsan 22 .
```

## 3.2.2 input

在上一节中，我们学习了字符串的拼接，现在我们来学习一下Python中的input()函数，它的作用是从标准输入中读取一行文本，并返回一个字符串。

我们新建一个名为input\_demo.py的文件，内容如下：

```
n=input("Enter any content")
print("Your input is"%r"%n
```

运行程序，按下F5，输入任意内容，按下回车，程序会输出你输入的内容。

### Python Shell

```
=====RESTART:D:/demo/input_demo.py=====
=====
```



Your input is 'Tom'

### 3.2.3 ☐ ☐ ☐ ☐ ☐

# Python Shell

□ □

# Python Shell

```
>>>print("'''''")
'''
>>>print('""')
""
>>>print""
SyntaxError:invalid character in identifier
```

Python #

## Python Shell

```
>>>#  
>>>print("hell world") #hello world  
hell world
```

xx.py

```
""  
"  
print("^_^"  
""  
...  
This is a  
Multi line comment  
...
```

## 3.3

Python

## 3.3.1 if

Python 的 if...else...

Python Shell

```
>>>a=2
>>>b=3
>>>if a>b:
    print("a max!")
else:
    print("b max!")
b max!
```

如果 a > b 为真，则执行 a > b 后面的代码，即打印 "a max!"，  
否则执行 "b max!"。

Python 的 if...else... 语句，  
可以嵌套使用，最多可以嵌套 4 层。

Python 的 if...else... 语句，  
可以嵌套使用，最多可以嵌套 4 层。

Python Shell

```
>>>student="xiaoming"
>>>if student=="xiaoming":
    print("xiaoming, You are on duty today.")
else:
    print("Please call xiaoming to duty")
```

xiaoming, You are on duty today.

```
Python"""!="Python"""if  
"in""not in"""
```

## Python Shell

```
>>>hi= "hello world"  
>>>if "hello" in hi:  
    print("Contain")  
else:  
    print("Not Contain")  
Contain
```

```
if"""
```

## Python Shell

```
>>>a=True  
>>>if a:  
    print("a is True")  
else:  
    print("a is not True")  
a is True
```

```
"""if"""
```

if\_demo.py

```
if results>=90:
    print('A')
elif results>=70:
    print('B')
elif results>=60:
    print('C')
else:
    print('D')
```

### 3.3.2 for

# Python Shell

# he

l

l

o

w

o

r

l

d

□□□□□□□□□□□□□□Python□□□□□□□□□□

## Python Shell

```
>>>fruits=['banana', 'apple', 'mango']
```

```
>>>for fruit in fruits:
```

```
    print(fruit)
```

banana

apple

mango

□□□□□□□□□□□□□□□□range()□□□

## Python Shell

```
>>>for i in range(5):
```

```
    print(i)
```

0  
1  
2  
3  
4

range() 함수는 시작, 끝, 스텝을 지정하여 값을 생성한다. 1, 10, 2를 지정하면 1부터 10까지의 값을 2씩 증가하여 생성한다.

## Python Shell

```
>>>for i in range(1,10,2):  
    print(i)
```

1  
3  
5  
7  
9

```
range(start,end[,step])
```

range() 함수는 start, end, step을 지정하여 값을 생성한다.

Python 2에서는 range() 대신 xrange()를 사용한다. Python 3에서는 range()만 사용한다.

## Python 2 xrange()

### 3.4 ☐ ☐ ☐ ☐ ☐

# Python

### 3.4.1 〇〇

[illegible]

# Python Shell

```
>>> lists=[1,2,3,'a',5]
>>> lists
[1, 2, 3, 'a', 5]
>>> lists[0]
1
>>> lists[4]
5
>>> lists[4]='b'
>>> lists[4]
'b'
>>> lists.append('c')
>>> lists
[1, 2, 3, 'a', 'b', 'c']
```



Python字典的初始化方法如下：  
lists[0].append()

## 3.4.2 字典

字典的初始化方法如下：  
字典的键值对(key-value)的表示方法为：key:value

### Python Shell

```
>>>dicts={"username": "zhangsan", 'password':123456}
>>>dicts.keys()
['username', 'password']
>>>dicts.values()
['zhangsan', 123456]
>>>dicts.items()
[('username', 'zhangsan'), ('password', 123456)]
>>>for k,v in dicts.items():
    print("dicts keys is"%k)
    print("dicts values is"%v)
```

```
dicts keys is 'username'
dicts values is 'zhangsan'
dicts keys is 'password'
dicts values is 123456
```

---

Python 字典的 key 和 value

`keys()` 返回字典的 key 列表，`values()` 返回字典的 value 列表，`items()` 返回字典的 key-value 列表。

zidian.py

```
#zip 函数返回 List 字典
```

```
#字典
```

```
keys=["b", "a", "c", "e", "d"]
```

```
values=["2", "1", "3", "5", "4"]
```

```
for key,value in zip(keys, values):
```

```
    print(key, value)
```

```
=====
```

```
=====RESTART:D:/demo/zidian.py=====
```

```
=====
```

```
b 2
```

```
a 1
```

```
c 3
```

```
e 5
```

```
d 4
```

## 3.5 字典的遍历

Python Shell  
Python Shell

## 3.5.1 Python Shell

Python Shell

Python Shell

```
>>>def add(a, b):
```

```
    print(a+b)
```

```
>>>add(3, 5)
```

```
8
```

Python Shell

```
def add(a, b):  
    print(a+b)  
add(3, 5)
```

```
def add(a, b):  
    return a+b
```

Python Shell

```
>>>def add(a, b):
```

```
    return a+b
```

```
>>>add(3, 5)
```

```
8
```

```
def add(a, b):  
    add(a, b)
```

# Python Shell

```
>>>def add(a=1, b=2):
    return a+b
```

```
>>>add()
```

3

```
>>>add(3,5)
```

8

```

add()

```

### 3.5.2 ☐ ☐ ☐ ☐

[illegible]

Python class

class\_test.py

```
class A(object):
```

```
def add(self, a, b):
    return a+b
```

```
count=A()
```

```
print(count.add(3, 5))
```

```
=====RESTART:D:/demo/class_test.py=====
```

```
class A:
    def __init__(self):
        self.add()
    def add(self):
        pass
```

```
init
```

```
return self.a+self.b
```

```
count=A('4', 5)
print(count.add())
```

□ □ □ □ □

```
=====RESTART:D:/demo/class_test.py=====
```

=====

9

```
class A:
    def __init__(self):
        self.a = 1
        self.b = 2
    def add(self):
        return self.a + self.b
a = A()
print(a.add())
```

# Python

class\_test.py

```
class A():
```

```
def add(self, a, b):
    return a+b
```

```
class B(A):
```

```
def sub(self, a, b):  
    return a-b
```



```
import time
```

```
print(time.ctime())
```

```
□□□□□
```

```
=====RESTART:D:/demo/imp.py=====
```

```
=====
```

```
Tue Dec 09 22:35:57 2014
```

```
□time□□□□□□ctime()□□□□□□□□□□print()□□□□□□□□□□  
□□□□□□□□□□time□□□□ctime()□□□□□□□□□□
```

imp.py

```
from time import ctime
```

```
print(ctime())
```

```
□□□□□
```

```
=====RESTART:D:/demo/imp.py=====
```

```
=====
```

```
Tue Dec 09 22:36:38 2014
```

```
□□□□□□□□□□Python□ctime()□□□time□□□□□□□□□□□□□□□□  
□□□time□□□□□sleep()□□□□□□□□□□□□□□□sleep()□□□□□□□□□□□  
□□□□□□□□□□□time□□□□□□□□□□□□□□□□
```

imp.py



```
from time import*
```

```
print(ctime())
```

```
print("####")
```

```
sleep(2)
```

```
print(ctime())
```

```
####
```

```
=====RESTART:D:/demo/imp.py=====
```

```
=====
```

```
Tue Dec 09 22:47:35 2014
```

```
####
```

```
Tue Dec 09 22:47:37 2014
```

```
    """#####time#####import#####
Python#####ctime#####
#####help()#####time#####
```

## Python Shell

```
>>>import time
```

```
>>>help(time)
```

```
Help on built-in module time:
```

### NAME

```
time-This module provides various functions to manipulate
time values.
```

## DESCRIPTION

There are two standard representations of time. One is the number

of seconds since the Epoch, in UTC (a.k.a. GMT). It may be an integer

or a floating point number (to represent fractions of seconds).

The Epoch is system-defined; on Unix, it is generally January 1st, 1970.

The actual value can be retrieved by calling `gmtime(0)`.

The other representation is a tuple of 9 integers giving local time.

The tuple items are:

year (including century, e.g. 1998)

month (1-12)

day (1-31)

hours (0-23)

minutes (0-59)

seconds (0-59)

weekday (0-6, Monday is 0)

Julian day (day in the year, 1-366)

DST (Daylight Savings Time) flag (-1, 0 or 1)

If the DST flag is 0, the time is given in the regular time zone;

if it is 1, the time is given in the DST time zone;

if it is -1, `mktime()` should guess based on the date and

time.

#### Variables:

timezone--difference in seconds between UTC and local standard time

altzone--difference in seconds between UTC and local DST time

daylight--whether local time should reflect DST

tzname--tuple of (standard time zone name, DST time zone name)

#### Functions:

time()--return current time in seconds since the Epoch as a float

clock()--return CPU time since process start as a float

sleep()--delay for a number of seconds given as a float

gmtime()--convert seconds since Epoch to UTC tuple

localtime()--convert seconds since Epoch to local time tuple

asctime()--convert time tuple to string

ctime()--convert time in seconds to string

mktime()--convert local time tuple to seconds since Epoch

strftime()--convert time tuple to string according to format specification

strptime()--parse string to time tuple according to format specification

```
tzset()--change the local timezone
```

.....

```
pip install Python selenium webdriver
Selenium
```

```
Python ..\Python35\Lib\site-
packages\Selenium\Selenium
Selenium
```

## 3.6.2

```


```

```
project
```

```
project/
```

```
├─ pub.py
```

```
└─ count.py
```

```
pub.py add
```

```
pub.py
```

```
def add(a, b):
```

```
    return a+b
```

```
count.py pub.py add()
```

```
print(add(4,5))
```

□ □ □ □ □

```
=====RESTART:D:/project/count.py=====
```

=====

9

□ □ □ □ □ □ □ □ □ □ □ □ □ □

```

    project
__pycache__/_pub.cpython-35.pyc

```

```

__pycache__
module.version.pyc version
Python CPython 3.5 pub.py
__pycache__/pub.cpython-35.pyc

```

### 3.6.3

[illegible]



□□□□□□

count.py

```
class A():  
  
    def add(self,a, b):  
        return a+b
```

new\_count.py

```
from count import A
```

```
class B(A):  
  
    def sub(self,a, b):  
        return a-b
```

```
resule=B().add(2, 5)  
print(resule)
```

□□□□

=====RESTART:D:/project/model/test.py=====

=====

7

□□□□□□Python 2□□Python 3□□new\_count.py□□□□□□□□  
□model□□□□□test.py□

```
1 module module module
```



2 module sys.module

3 module

4 module

3 module

sys.path PythonPATH Python  
PythonPATH  
module module module  
xml.py import xml module  
xml

package module  
package module Python 2 package  
\_\_init\_\_.py

new\_count.py “from  
count import A” “count”  
test.py “from count import A” “count”

“from .count import A” count  
. test.py count new\_count.py  
test.py

new\_count.py



test.py

```
import sys
sys.path.append("./model") #model의 경로에 path
from model import new_count

test=new_count.B()
test.add(2,5)
```

실행

=====RESTART:D:/project/test.py=====

=====

7

Python 2.../model/model\_\_init\_\_.py  
Python

## 3.7

Python exception object  
Traceback

“”  
Traceback

### 3.7.1

Python Shell

## Python Shell

```
>>>open("abc.txt", 'r')
```

```
Traceback (most recent call last):
```

```
File"<pyshell#0>", line 1, in<module>
```

```
    open("abc.txt", 'r')
```

```
FileNotFoundError: [Errno 2] No such file or directory: 'abc.txt'
```

Python Shell  
FileNotFoundError: [Errno 2] No such file or directory: 'abc.txt'  
Python Shell

Python Shell  
FileNotFoundError: [Errno 2] No such file or directory: 'abc.txt'  
Python Shell

## abnormal.py

```
try:
```

```
    open("abc.txt", 'r')
```

```
except FileNotFoundError:
```

```
    print("Error!")
```

```
=====
```

```
=====RESTART: D:\demo\abnormal.py=====
```

```
=====
```

```
Error!
```

```
        except FileNotFoundError:
            print("File not found")
```

abnormal.py

```
try:
    print(aa)
except FileNotFoundError:
    print("File not found")
```

Python

```
=====RESTART:D:\demo\abnormal.py=====
=====
```

```
Traceback (most recent call last):
  File "D:\demo\abnormal.py", line 2, in <module>
    print(aa)
NameError: name 'aa' is not defined
```

```
        except:
            print("File not found")
        except NameError:
            except FileNotFoundError:
                print("File not found")
```

Python

abnormal.py

```
try:
    print(aa)
```

```
except NameError:
    print("name!")
```

=====RESTART:D:\demo\abnormal.py=====

name!

1 handler

2

3 "main"

Traceback

Python Exception

abnormal.py

```
try:
    open("abc.txt", 'r')
except Exception:
    print("!!!")
```

!!!!

=====RESTART:D:\demo\abnormal.py=====

!!!

Python 2.5 BaseException  
ExceptionBaseExceptionBaseException  
!!!!

abnormal.py

```
try:
    open("abc.txt", 'r')
    print(aa)
except BaseException:
    print("!!!")
```

!!!!

=====RESTART:D:\demo\abnormal.py=====

=====

!!!

print() Python

abnormal.py

```
try:
    open("abc.txt", 'r')
    print(aa)
except BaseException as msg:
    print(msg)
```

=====

=====RESTART:D:\demo\abnormal.py=====

=====

[Errno 2] No such file or directory:'abc.txt'

BaseException msg print

Python 2 Python 2 "as"

Python 3.1

### 3.1 Python

|  |  |
|--|--|
|  |  |
|--|--|



| 名前                | 説明                          |
|-------------------|-----------------------------|
| BaseException     | 例外の基底クラス                    |
| Exception         | BaseExceptionのサブクラス         |
| AssertionError    | assert文で失敗した場合              |
| FileNotFoundError | ファイルが見つからない場合               |
| AttributeError    | 属性が存在しない場合                  |
| OSError           | OS関連のエラー、例として"/O"や"O"などのエラー |
| NameError         | 変数が定義されていない場合               |
| IndexError        | インデックスが範囲外の場合               |
| SyntaxError       | 構文エラー                       |
| KeyboardInterrupt | Ctrl+Cで中断された場合              |
| TypeError         | 型エラー                        |

## 3.7.2 例外処理

try...except  
else

abnormal.py

```

try:
    aa="aaaaa"
    print(aa)
except Exception as msg:
    print(msg)
else:
    print("aaaa!")

```

aaaa

=====RESTART:D:\demo\abnormal.py=====

=====

aaaa

aaaa

aaaaaaaelseelse  
 aaaaaaa  
 aaaaaaa try...except...finally...aaaaaa  
 aaaa

abnormal.py

```

try:
    print(aa)
except Exception as e:
    print(e)
finally:
    print("aaaaaaaaaaaa")

```

□□□□

=====RESTART:D:\demo\abnormal.py=====

=====

name 'aa' is not defined

□□□□□□□□□□

□□□□□□□□aa□□

abnormal.py

try:

aa="□□□□"

print(aa)

except Exception as e:

print(e)

finally:

print("□□□□□□□□□□□□□□")

□□□□

=====RESTART:D:\demo\abnormal.py=====

=====

□□□□

□□□□□□□□□□

□□□□□□□□□□□□□□finally□□□□□□

### 3.7.3 □□□□

```
print()Pythonraise
raise
```

abnormal.py

```
from random import randint
```

```
#19
```

```
number=randint(1,9)
```

```
if number%2==0:
```

```
    raise NameError("%d is even"%number)
```

```
else:
```

```
    raise NameError("%d is odd"%number)
```

```
=====RESTART:D:\demo\abnormal.py=====
```

```
=====
```

```
Traceback (most recent call last):
```

```
File"D:\project\count.py", line 8, in<module>
```

```
    raise NameError("%d is even"%number)
```

```
NameError:4 is even
```

```
randint()19
```

```
raiseNameErrorNameError
```

```
raise
```

raise Python abcError  
Python abcError

Python  
Python Python

1 Python Python  
C Python Python  
path Python

2 D:\xx\test case  
list\test.py IDE Sublime Text

3 D:\selenium\webdriver.py  
Python



## 第4章 WebDriver API

---

WebDriver API 是 Selenium 框架中 WebDriver 的核心 API。本章将介绍 WebDriver API 的基本用法，包括如何创建 WebDriver 实例、如何操作元素、如何执行 JavaScript 代码等。本章还将介绍 WebDriver 的一些高级用法，如如何设置 WebDriver 的选项、如何执行 WebDriver 的异步操作等。

本章将介绍 WebDriver API 的基本用法，包括如何创建 WebDriver 实例、如何操作元素、如何执行 JavaScript 代码等。

### 4.1 创建 WebDriver 实例

创建 WebDriver 实例是使用 WebDriver API 的第一步。本章将介绍如何创建 WebDriver 实例，包括如何安装 WebDriver 的依赖包、如何创建 WebDriver 实例等。

创建 WebDriver 实例需要使用 WebDriver 的工厂方法。WebDriver 的工厂方法会根据传入的参数创建不同类型的 WebDriver 实例。本章将介绍如何创建不同类型的 WebDriver 实例，包括如何创建 Chrome WebDriver 实例、如何创建 Firefox WebDriver 实例等。

创建 WebDriver 实例时，可以传入一些选项来配置 WebDriver 的行为。本章将介绍如何配置 WebDriver 的选项，包括如何设置 WebDriver 的浏览器版本、如何设置 WebDriver 的代理服务器等。



#### 4.1 Web

4.2 WebDriver



#### 4.2 FireBug

WebDriver Python



- id → name
- class name → tag name
- link text → partial link text
- xpath → css selector
- find\_element\_by\_id() →  
find\_element\_by\_name()
- find\_element\_by\_class\_name() →  
find\_element\_by\_tag\_name()
- find\_element\_by\_link\_text() →  
find\_element\_by\_partial\_link\_text()
- find\_element\_by\_xpath() →  
find\_element\_by\_css\_selector()

1. 爬取百度首页的链接  
 2. 爬取百度首页的标题

baidu.htmlindex.

```

<html>
<head><body>
<script>
  <div id="wrapper" style="display:block;">
    <div id="debug" style="display:block;position:...">
      <script>
        <div id="head" class="s_down">
          <div class="head_wrapper">

```

```

<div class="s_form">
  <div class="s_form_wrapper">
    <div id="lg">
      <a id="result_logo" onmousedown="return .."
href="/">
        <form id="form" class="fm" action="/s"
name="f">
          <input type="hidden" value="utf-8"
name="ie">
            <input type="hidden" value="8" name="f">
              <input type="hidden" value="1"
name="rsv_bp">
                <input type="hidden" value="1"
name="rsv_idx">
                  <input type="hidden" value="" name="ch">
                  <input type="hidden" value="02.." name="tn">
                  <input type="hidden" value="" name="bar">
                  <span class="bg s_ipt_wr">
                    <input id="kw" class="s_ipt"
autocomplete="off"
                      maxlength="100" value="" name="wd">
                  </span>
                  <span class="bg s_btn_wr">
                    <input id="su" class="bg s_btn
"type="submit"
                      value="□□□□">
                  </span>

```

....

</body>

</html>hello

HTML 文档的骨架

① 文档的骨架

```
<html></html>
```

```
<body></body>
```

```
<div></div>
```

```
<form></form>
```

html 文档的骨架

② 文档的骨架

```
<div id="head" class="s_down">
```

```
<form class="well">
```

```
<input id="kw" name="wd" class="s_ipt">
```

HTML 文档的骨架

③ 文档的骨架

```
<a>  </a>
<a>hao123</a>
<a>  </a>
```

```
<a>  </a>
<a>hao123</a>
<a>  </a>
```

```
<a>  </a>
<a>hao123</a>
<a>  </a>
```

④ □□□□□□

```
<html>
  <body>
    </body>
</html>
<div>
  <from>
    <input/>
  </from>
</div>
```

```
<html>
  <body>
    </body>
</html>
<div>
  <from>
    <input/>
  </from>
</div>
```

```
<html>
  <body>
    </body>
</html>
<div>
  <from>
    <input/>
  </from>
</div>
```

```
<html>
  <body>
    </body>
  </html>
  <div>
    <from>
      <input/>
    </from>
  <div>
```

```
<html>
  <body>
    </body>
  </html>
  <div>
    <from>
      <input/>
    </from>
  <div>
```

```
<html>
  <body>
    </body>
  </html>
  <div>
    <from>
      <input/>
    </from>
  <div>
```

```
<html>
  <body>
    </body>
  </html>
  <div>
    <from>
      <input/>
    </from>
  <div>
```

```
<html>
  <body>
    </body>
  </html>
  <div>
    <from>
      <input/>
    </from>
  <div>
```

```
<html>
  <body>
    </body>
  </html>
  <div>
    <from>
      <input/>
    </from>
  <div>
```

inputform[illegible]

```
.....  
<input id="kw" class="s_ipt" autocomplete="off" maxlength="100"  
value=""  
name="wd">  
  
.....  
<input id="su" class="bg s_btn" type="submit" value="□□□□">  
  
.....
```

```
.....  
<input id="kw" class="s_ipt" autocomplete="off" maxlength="100"  
value=""  
name="wd">  
  
.....  
<input id="su" class="bg s_btn" type="submit" value="□□□□">  
  
.....
```

```
.....  
<input id="kw" class="s_ipt" autocomplete="off" maxlength="100"  
value=""  
name="wd">  
  
.....  
<input id="su" class="bg s_btn" type="submit" value="□□□□">  
  
.....
```

```
.....  
<input id="kw" class="s_ipt" autocomplete="off" maxlength="100"  
value=""  
name="wd">  
  
.....  
  
<input id="su" class="bg s_btn" type="submit" value="□□□□">  
  
.....
```

```

.....
<input id="su" class="bg s_btn" type="submit" value="□□□□">
.....

```

```

.....
<input id="su" class="bg s_btn" type="submit" value="□□□□">
.....

```

WebDriver.findElement()

WebDriver.findElement() 的查找策略有 4 种：  
id、name、class name、tag name

除了这 4 种，还可以使用 XPath 和 CSS 选择器来查找元素。

WebDriver.findElement() 的查找策略有 4 种：  
id、name、class name、tag name、XPath、CSS 选择器。

WebDriver.findElement() 的查找策略有 4 种：

## 4.1.1 id

HTML 元素的 id 属性是 HTML 元素的一个属性，用于唯一标识该元素。  
WebDriver.findElement() 的 id 属性查找策略，就是通过元素的 id 属性来查找元素。

```
find_element_by_id("kw")  
find_element_by_id("su")
```

find\_element\_by\_id() 的 id 属性查找策略

## 4.1.2 name

HTML의 name 속성을 가진 요소의 name 값을 반환합니다.  
name 속성을 가진 요소의 name 값을 반환합니다.

```
find_element_by_name("wd")
```

find\_element\_by\_name()는 name 속성을 가진 요소의 name 값을 반환합니다.  
name 속성을 가진 요소의 name 값을 반환합니다.

## 4.1.3 class

HTML의 class 속성을 가진 요소의 id, name, class 값을 반환합니다.  
class 속성을 가진 요소의 class 값을 반환합니다.

```
find_element_by_class_name("s_ip")
```

```
find_element_by_class_name("bg s_btn")
```

```
find_element_by_class_name()는 class 속성을 가진 요소의 class 값을 반환합니다.
```

## 4.1.4 tag

HTML의 tag 속성을 가진 요소의 tag 값을 반환합니다.  
tag 속성을 가진 요소의 tag 값을 반환합니다.  
<div> <input> <a> tag의 tag name 값을 반환합니다.  
tag

```
tag name 속성을 가진 요소의 tag name 값을 반환합니다.
```

```
find_element_by_tag_name("input")
```

```
find_element_by_tag_name() 返回tag name
```

## 4.1.5 link

link 返回一个包含所有链接的列表，每个链接都是一个字典，包含以下键：

```
<a      class="mnav"      name="tj_trnews"
href="http://news.baidu.com"></a>
      <a      class="mnav"      name="tj_trhao123"
href="http://www.hao123.com">hao123</a>
      <a      class="mnav"      name="tj_trmap"
href="http://map.baidu.com"></a>
      <a      class="mnav"      name="tj_trvideo"
href="http://v.baidu.com"></a>
      <a      class="mnav"      name="tj_trtieba"
href="http://tieba.baidu.com"></a>
```

字典中的键包括：name、href、text、link\_text、link\_attribute、link\_attribute\_names、link\_attribute\_values、link\_attribute\_types、link\_attribute\_types\_names、link\_attribute\_types\_values、link\_attribute\_types\_names\_values。

```
find_element_by_link_text("链接")
```

```
find_element_by_link_text("hao123")
```

```
find_element_by_link_text("")  
  
find_element_by_link_text("")  
  
find_element_by_link_text("")  
  
find_element_by_link_text())
```

```
find_element_by_link_text("")  
  
find_element_by_link_text("")  
  
find_element_by_link_text("")  
  
find_element_by_link_text())
```

```
find_element_by_link_text("")  
  
find_element_by_link_text("")  
  
find_element_by_link_text("")  
  
find_element_by_link_text())
```

```
find_element_by_link_text("")  
  
find_element_by_link_text("")  
  
find_element_by_link_text("")  
  
find_element_by_link_text())
```

### 4.1.6 partial link

[illegible]

<a class="mnav" name="tj\_lang" href="#">□□□□□□□□□□</a>

partial link

```
find_element_by_partial_link_text("aaaaa")

find_element_by_partial_link_text("aaaa")
```

```
find_element_by_partial_link_text("aaaaa")

find_element_by_partial_link_text("aaaa")
```

```
find_element_by_partial_link_text() [] [] [] [] [] [] [] [] [] [] [] []
[] [] [] []
```

```
    id  
name  
  
id name  
id name
```





div[2]div

XPath

```
find_element_by_xpath("//input[@id='kw']")
```

```
find_element_by_xpath("//input[@id='su']")
```

//input[@id='kw']id  
kwnameclass

```
find_element_by_xpath("//input[@name='wd']")
```

```
find_element_by_xpath("//input[@class='s_ip']")
```

```
find_element_by_xpath("//*[@class='bg_s_btn']")
```

\*XPathidname  
class

```
find_element_by_xpath("//input[@maxlength='100']")
```

```
find_element_by_xpath("//input[@autocomplete='off']")
```

```
find_element_by_xpath("//input[@type='submit']")
```

|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|

`baidu.html`

■■■■■

```
<form id= "form" class= "fm" action= "/s" name= "f">
  <input type= "hidden" value= "utf-8" name= "ie">
  <input type= "hidden" value= "8" name= "f">
  <input type= "hidden" value= "1" name= "rsv_bp">
  <input type= "hidden" value= "1" name= "rsv_idx">
  <input type= "hidden" value= "" name= "ch">
  <input type= "hidden" value= "02.." name= "tn">
  <input type= "hidden" value= "" name= "bar">
  <span class= "bg s_ipt_wr">
    <input id= "kw" class= "s_ipt" autocomplete= "off"
      maxlength= "100" value= "" name=
"wd">
  </span>
  <span class= "bg s_btn_wr">
    <input id= "su" class= "bg s_btn" type= "submit"
      value= "□□□□">
  </span>
```

■■■■■

[illegible]

XPath

```
find_element_by_xpath("//span[@class='bg s_ipt_wr']/input")
```

span[@class='bg s\_ipt\_wr']class

```
find_element_by_xpath("//form[@id='form']/span/input")
```

```
find_element_by_xpath("//form[@id='form']/span[2]/input")
```

<html>

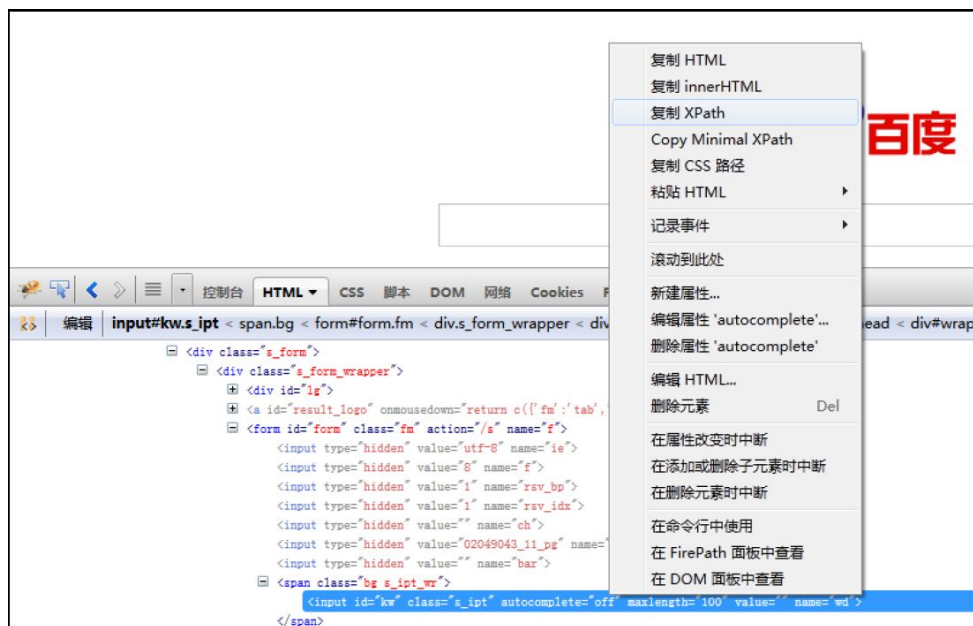
```
.....  
<input id="kw" class="su" name="ie">  
<input id="kw" class="aa" name="ie">  
<input id="bb" class="su" name="ie">  
.....
```

id  
class  
idclass  
"and"

```
find_element_by_xpath("//input[@id='kw' and@class='su']/span/input")
```

“and”

1FirebugFirePathXPathFirefoxFireBugXPath4.3



4.3 FireBugXPath

FirePathXPathXPath4.4



图4.4 使用FirePath工具XPath

# 4.1.8 CSS

CSS(Cascading Style Sheets)是用于描述HTML/XML文档外观的标记语言。CSS是Selenium WebDriver中用于定位元素的重要工具。

CSS选择器是用于定位HTML/XML文档中元素的一种方式。XPath选择器是另一种定位元素的方式。CSS选择器通常比XPath选择器更简洁、更高效。

## CSS选择器

### 4.1 CSS选择器

| 选择器    | 定位         | 说明                           |
|--------|------------|------------------------------|
| .class | .intro     | class选择器<br>class="intro"的元素 |
| #id    | #firstname | id选择器<br>id="firstname"的元素   |



```
find_element_by_css_selector(".bg s_btn")
```

find\_element\_by\_css\_selector() CSS 클래스 선택자. 클래스  
class 선택자

2 id 선택자

```
find_element_by_css_selector("#kw")
```

```
find_element_by_css_selector("#su")
```

3 # id 선택자

선택자

```
find_element_by_css_selector("input")
```

CSS 선택자. CSS 선택자는 HTML 문서에서 특정 요소를 선택하는 데 사용됩니다. CSS 선택자는 HTML 요소를 선택하는 데 사용됩니다.

1 선택자

```
find_element_by_css_selector("span>input")
```

선택자. span 선택자. input 선택자

2 선택자

```
find_element_by_css_selector("[autocomplete=off]")
```

```
find_element_by_css_selector("[name='kw']")
```



```
find_element_by_css_selector('[type="submit"]')
```

CSS selector 是 CSS 选择器，用于在 HTML 文档中查找元素。它由一个或多个选择器组成，用于定位文档中的元素。选择器可以是类选择器、ID 选择器、元素选择器、属性选择器等。在 Selenium 中，使用 `find_element_by_css_selector()` 方法可以找到指定的元素。

3. 提交按钮

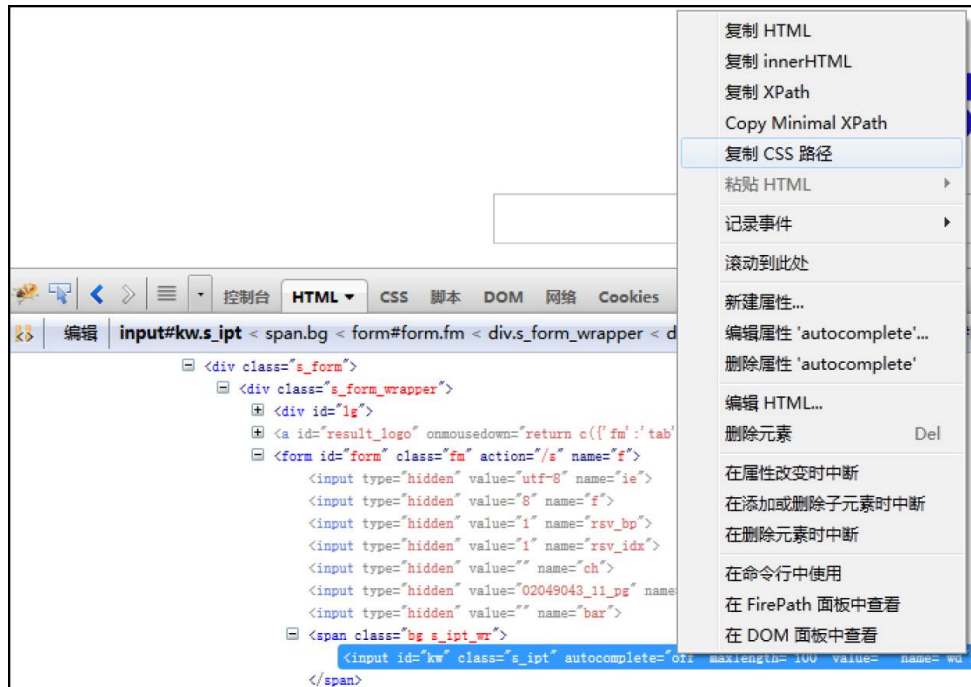
提交按钮的 CSS 选择器为 `span.bg s_btn_wr>input#su`

```
find_element_by_css_selector("span.bg s_ipt_wr>input.s_ipt")
```

```
find_element_by_css_selector("span.bg s_btn_wr>input#su")
```

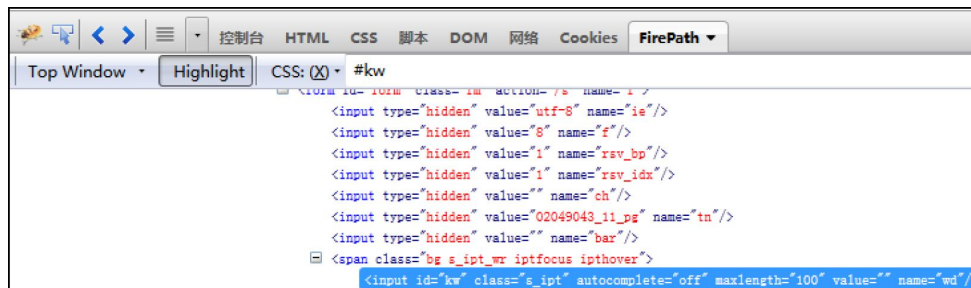
提交按钮的 CSS 选择器为 `span[class=bg s_ipt_wr]>input[class=s_ipt]`。在 Selenium 中，使用 `find_element_by_css_selector()` 方法可以找到指定的元素。

Firebug 是 Firefox 浏览器中的一个插件，用于查看网页的 HTML 和 CSS 代码。在 Firebug 的 CSS 面板中，可以找到元素的 CSS 选择器。在 Firebug 的 CSS 面板中，可以找到元素的 CSS 选择器。在 Firebug 的 CSS 面板中，可以找到元素的 CSS 选择器。



#### 4.5 使用Firebug复制CSS

### 使用FirePath复制CSS 4.6



#### 4.6 使用FirePath复制CSS

学习CSS的教程很多，其中W3CSchool  
 是一个不错的选择。

### XPath与CSS 4.2

#### 4.2 XPath与CSS

|  |  |  |
|--|--|--|
|  |  |  |
|--|--|--|

| Locator         | XPath                            | CSS  |
|-----------------|----------------------------------|--|
| Tag             | //div                            | div  |
| <b>By id</b>    | //div[@id='eleid']               | div#eleid  |
| <b>By class</b> | //div[@class='eleclass']         | div.eleid  |
| <b>By text</b>  | //div[@title='Move mouse here']  | div[title=Move mouse here]<br>div[title^=Move]<br>div[title\$=here]<br>div[title*=mouse] |
| Parent/Child    | //div[@id='eleid']/*<br>//div/h1 | div#eleid>*<br>div>h1  |

XPath and CSS selectors are used to locate elements on a web page. XPath is a language for navigating through an XML document, while CSS is a style sheet language. Both are used to identify elements in a web page for automation.

Web automation frameworks like Selenium use these locators to interact with web elements. WebDriver is a tool that allows you to control a web browser programmatically.

## 4.1.9 By text

In Selenium, you can locate an element by its text content using the `By.text()` locator. This is useful when you know the exact text of the element you want to interact with.

```
find_element(By.ID, "kw")

find_element(By.NAME, "wd")

find_element(By.CLASS_NAME, "s_ipt")

find_element(By.TAG_NAME, "input")

find_element(By.LINK_TEXT, "")

find_element(By.PARTIAL_LINK_TEXT, "")

find_element(By.XPATH, "//*[@class='bg s_btn']")

find_element(By.CSS_SELECTOR, "span.bg s_btn_wr>input#su")
```

```
find_element(By.ID, "kw")

find_element(By.NAME, "wd")

find_element(By.CLASS_NAME, "s_ipt")

find_element(By.TAG_NAME, "input")

find_element(By.LINK_TEXT, "")

find_element(By.PARTIAL_LINK_TEXT, "")

find_element(By.XPATH, "//*[@class='bg s_btn']")

find_element(By.CSS_SELECTOR, "span.bg s_btn_wr>input#su")
```

```
find_element(By.ID, "kw")

find_element(By.NAME, "wd")

find_element(By.CLASS_NAME, "s_ipt")

find_element(By.TAG_NAME, "input")

find_element(By.LINK_TEXT, "")

find_element(By.PARTIAL_LINK_TEXT, "")

find_element(By.XPATH, "//*[@class='bg s_btn']")

find_element(By.CSS_SELECTOR, "span.bg s_btn_wr>input#su")
```

```
find_element(By.ID, "kw")

find_element(By.NAME, "wd")

find_element(By.CLASS_NAME, "s_ipt")

find_element(By.TAG_NAME, "input")

find_element(By.LINK_TEXT, "")

find_element(By.PARTIAL_LINK_TEXT, "")

find_element(By.XPATH, "//*[@class='bg s_btn']")

find_element(By.CSS_SELECTOR, "span.bg s_btn_wr>input#su")
```

```
find_element(By.ID, "kw")

find_element(By.NAME, "wd")

find_element(By.CLASS_NAME, "s_ipt")

find_element(By.TAG_NAME, "input")

find_element(By.LINK_TEXT, "")

find_element(By.PARTIAL_LINK_TEXT, "")

find_element(By.XPATH, "//*[@class='bg s_btn']")

find_element(By.CSS_SELECTOR, "span.bg s_btn_wr>input#su")
```

```
find_element(By.ID, "kw")

find_element(By.NAME, "wd")

find_element(By.CLASS_NAME, "s_ipt")

find_element(By.TAG_NAME, "input")

find_element(By.LINK_TEXT, "")

find_element(By.PARTIAL_LINK_TEXT, "")

find_element(By.XPATH, "//*[@class='bg s_btn']")

find_element(By.CSS_SELECTOR, "span.bg s_btn_wr>input#su")
```

```
find_element(By.ID, "kw")

find_element(By.NAME, "wd")

find_element(By.CLASS_NAME, "s_ipt")

find_element(By.TAG_NAME, "input")

find_element(By.LINK_TEXT, "")

find_element(By.PARTIAL_LINK_TEXT, "")

find_element(By.XPATH, "//*[@class='bg s_btn']")

find_element(By.CSS_SELECTOR, "span.bg s_btn_wr>input#su")
```

```
find_element(By.ID, "kw")

find_element(By.NAME, "wd")

find_element(By.CLASS_NAME, "s_ipt")

find_element(By.TAG_NAME, "input")

find_element(By.LINK_TEXT, "")

find_element(By.PARTIAL_LINK_TEXT, "")

find_element(By.XPATH, "//*[@class='bg s_btn']")

find_element(By.CSS_SELECTOR, "span.bg s_btn_wr>input#su")
```

```
find_element() By By
```

```
from selenium.webdriver.common.by import By
```

```
WebDriver.findElementById()
```

webelement.py

```
def find_element_by_id(self, id_):
```

```
"""Finds element within this element's children by ID.
```

```
:Args:
```

```
    -id_-ID of child element to locate.
```

```
"""
```

```
    return self.find_element(by=By.ID, value=id_)
```

```
.....
```

```
WebDriver
```

## 4.2

WebDriver

### 4.2.1

WebDriver (480\*800) set\_window\_size()

test.py

```
from selenium import webdriver
driver=webdriver.Firefox()
driver.get("http://m.mail.10086.cn")
```

```
#窗口大小
print("窗口大小480*800")
driver.set_window_size(480, 800)
driver.quit()
```

在PC上运行程序时，窗口大小默认为最大化。在代码中，我们使用 `maximize_window()` 方法将窗口最大化，使用 `set_window_size()` 方法将窗口大小设置为指定的宽度和高度。

## 4.2.2 WebDriver 操作

WebDriver 提供了许多方法来操作浏览器，包括 `back()` 和 `forward()` 方法，用于在浏览器的历史记录中前进和后退。此外，还有 `get()` 方法用于加载指定的 URL。

test.py

```
from selenium import webdriver

driver=webdriver.Firefox()

#访问百度
first_url='http://www.baidu.com'
print("now access%s"%(first_url))
driver.get(first_url)
```

```
#访问二级页面
second_url='http://news.baidu.com'
print("now access%s"%(second_url))
driver.get(second_url)
```

```
#返回一级页面
print("back to%s"%(first_url))
driver.back()
```

```
#访问二级页面
print("forward to%s"%(second_url))
driver.forward()
driver.quit()
```

print()输出URL

## Python shell

=====RESTART:D:/pyse/test.py=====

=====

```
now access http://www.baidu.com
now access http://news.baidu.com
back to http://www.baidu.com
forward to http://news.baidu.com
```

## 4.2.3 断言

songyaping\_1... F5 4.7 广播 36717 收听 13 听众 3

你在做什么呢？

☐ 同步到论坛

4.7

test.py

.....

driver.refresh() #

.....

## 4.3

WebDriver

- clear()
- send\_keys(\*value)
- click()

### 4.3.1 126





## 4.3.2 WebElement

WebElement 4.1 8  
3 WebElement

```
submit()
```

submit() “”  
submit()

youdao.py

```
from selenium import webdriver
```

```
driver=webdriver.Firefox()
```

```
driver.get("http://www.youdao.com")
```

```
driver.find_element_by_id('query').send_keys('hello')
```

```
#
```

```
driver.find_element_by_id('query').submit()
```

```
driver.quit()
```

submit() “”  
submit() click() submit()  
submit() click()

- size 元素尺寸
- text 元素内容
- get\_attribute(name) 获取属性
- is\_displayed() 是否显示

baidu.py

```
from selenium import webdriver
```

```
driver=webdriver.Firefox()
```

```
driver.get("http://www.baidu.com")
```

```
# 元素尺寸
```

```
size=driver.find_element_by_id('kw').size
```

```
print(size)
```

```
# 元素内容
```

```
text=driver.find_element_by_id("cp").text
```

```
print(text)
```

```
# 获取属性 id name type
```

```
attribute=driver.find_element_by_id("kw").get_attribute('type')
```

```
print(attribute)
```

```
# 是否显示 True False
```

```
result=driver.find_element_by_id("kw").is_displayed()
```

```
print(result)
```

□□□□

=====

©2015 Baidu 百度 京ICP030173

True

WebElement WebDriverAPI

## 4.4 ☐ ☐ ☐ ☐

## ActionChains

- perform() `Actions`

- context\_click() 单击
- double\_click() 双击
- drag\_and\_drop() 拖拽
- move\_to\_element() 移动到元素

4.8 文件管理

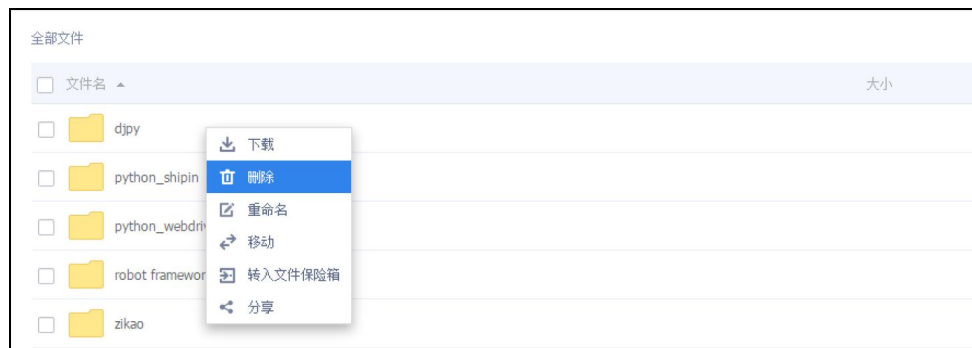


图4.8 文件管理

## 1. 环境搭建

使用ActionChains类可以实现链式调用，如click()方法。4.8节介绍了360文件管理器的使用。

yunpan.py

```
from selenium import webdriver
# 使用ActionChains
from selenium.webdriver.common.action_chains import ActionChains

driver=webdriver.Firefox()
```

```
driver.get("http://yunpan.360.cn")
```

```
#.....
```

```
#定位元素
```

```
right_click=driver.find_element_by_id("xx")
```

```
#定位元素
```

```
ActionChains(driver).context_click(right_click).perform()
```

```
#.....
```

- from selenium.webdriver import ActionChains  
引入ActionChains
- ActionChains(driver)  
创建ActionChains对象，传入driver
- context\_click(right\_click)  
context\_click()方法用于定位元素并单击
- perform()  
执行ActionChains对象中的操作

## 2. 定位元素

定位元素是 Selenium 4.9 版本



图4.9 百度搜索

`move_to_element()` 和 `context_click()` 方法

mouse.py

#.....

# 百度搜索

`above=driver.find_element_by_id("id")`

# 百度搜索

`ActionChains(driver).move_to_element(above).perform()`

#.....

### 3. 双击

`double_click` 方法

mouse.py

```
#.....
```

```
#원본요소찾기
```

```
double_click=driver.find_element_by_id("xx")
```

```
#복사할요소찾기
```

```
ActionChains(driver).double_click(double_click).perform()
```

```
#.....
```

## 4. 드래그 앤 드롭

```
drag_and_drop(source, target)은 원본요소와 복사할 요소를 지정하여  
원본에서 복사한 후 지정된 위치로 이동시킵니다.
```

- source: 원본요소
- target: 복사할 요소

mouse.py

```
#.....
```

```
#원본요소찾기
```

```
element=driver.find_element_by_id("xx")
```

```
#복사할요소찾기
```

```
target=driver.find_element_by_id("xx")
```

```
#원본요소찾기
```



```
ActionChains(driver).drag_and_drop(element, target).perform()  
#.....
```

## 4.5 键盘操作

Keys() 类包含所有键盘按键的常量，send\_keys() 方法可以发送指定的按键序列。例如，发送 Ctrl+A 和 Ctrl+C 的序列：

baidu.py

```
from selenium import webdriver  
# 导入 Keys 类  
from selenium.webdriver.common.keys import Keys  
  
driver=webdriver.Firefox()  
driver.get("http://www.baidu.com")  
  
# 定位搜索框  
driver.find_element_by_id("kw").send_keys("selenium")  
  
# 按下退格键  
driver.find_element_by_id("kw").send_keys(Keys.BACK_SPACE)  
  
# 按下空格键  
driver.find_element_by_id("kw").send_keys(Keys.SPACE)  
driver.find_element_by_id("kw").send_keys(" ")
```

```

# ctrl+a
driver.find_element_by_id("kw").send_keys(Keys.CONTROL, 'a')

# ctrl+x
driver.find_element_by_id("kw").send_keys(Keys.CONTROL, 'x')

# ctrl+v
driver.find_element_by_id("kw").send_keys(Keys.CONTROL, 'v')

#
driver.find_element_by_id("su").send_keys(Keys.ENTER)

driver.quit()

```

```

from selenium.webdriver.common.keys import Keys

```

```

keys

```

```


```

|                              |           |
|------------------------------|-----------|
| send_keys(Keys.BACK_SPACE)   | BackSpace |
| send_keys(Keys.SPACE)        | (Space)   |
| send_keys(Keys.TAB)          | (Tab)     |
| send_keys(Keys.ESCAPE)       | Esc       |
| send_keys(Keys.ENTER)        | Enter     |
| send_keys(Keys.CONTROL, 'a') | Ctrl+A    |

|   |          |
|---|----------|
| <code>send_keys(Keys.CONTROL, 'c')</code> | ⌨️Ctrl+C |
| <code>send_keys(Keys.CONTROL, 'x')</code> | ⌨️Ctrl+X |
| <code>send_keys(Keys.CONTROL, 'v')</code> | ⌨️Ctrl+V |
| <code>send_keys(Keys.F1)</code>           | ⌨️F1     |
| <code>send_keys(Keys.F12)</code>          | ⌨️F12    |

## 4.6 断点调试

断点调试是开发中非常常用的一个功能，可以帮助我们快速定位程序中的问题。在 Selenium 中，我们可以通过 `driver.pause()` 方法来暂停当前的操作，然后使用 `driver.get_log('debug')` 来查看当前的日志信息。通过这种方式，我们可以快速定位到程序中的问题，并进行相应的修复。

在断点调试的过程中，我们需要注意以下几点：首先，我们需要确保程序已经运行到了断点位置；其次，我们需要查看当前的日志信息，以确定问题的原因；最后，我们需要根据日志信息来修复程序中的问题。

在 Selenium 中，我们可以通过 `driver.pause()` 方法来暂停当前的操作，然后使用 `driver.get_log('debug')` 来查看当前的日志信息。通过这种方式，我们可以快速定位到程序中的问题，并进行相应的修复。

126 断点调试

login126.py

```
from selenium import webdriver
import time
```

```
driver=webdriver.Firefox()
driver.get("http://www.126.com")

print('Before login=====')
```

```
#获取title
title=driver.title
print(title)
```

```
#获取URL
now_url=driver.current_url
print(now_url)
```

```
#登录
driver.find_element_by_id("idInput").clear()
driver.find_element_by_id("idInput").send_keys("username")
driver.find_element_by_id("pwdInput").clear()
driver.find_element_by_id("pwdInput").send_keys("password")
driver.find_element_by_id("loginBtn").click()
time.sleep(5)
```

```
print('After login=====')
```

```
#获取title
title=driver.title
print(title)
```

```
#获取当前URL
```

```
now_url=driver.current_url
```

```
print(now_url)
```

```
#获取用户名
```

```
user=driver.find_element_by_id('spnUid').text
```

```
print(user)
```

```
driver.quit()
```

```
print('测试成功')
```

## Python Shell

```
=====RESTART:D:/pyse/login126.py=====
```

```
=====
```

```
Before login=====
```

```
126[1] - - [1]
```

```
http://www.126.com/
```

```
After login=====
```

```
[1] 6.0
```

```
http://mail.126.com/js6/main.jsp?
```

```
sid=VBqseScE0CvclcjRdjEEYbWiQFuWVamg[1]df=ma
```

```
il126_letter#module=welcome.WelcomeModule%7C%7B%7D
```

```
username@126.com
```

```
title[1]
```

current\_url返回当前URL

返回当前页面的title  
URL返回当前页面的URL  
title返回当前页面的title  
text返回当前页面的text  
username@126.com返回当前页面的username

## 4.7 异常处理

Webdriver的AJAX请求可能会导致ElementNotVisibleException异常  
ElementNotVisibleException异常通常是由于元素在页面加载过程中被隐藏或移除  
导致无法找到元素而引发的异常

WebDriverException异常

### 4.7.1 异常处理

WebDriverException异常  
TimeoutException异常

baidu.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as
```

EC

```
driver = webdriver.Firefox()
driver.get("http://www.baidu.com")
element = WebDriverWait(driver, 5, 0.5).until(
    EC.presence_of_element_located((By.ID,
    "kw")))
element.send_keys('selenium')
driver.quit()
```

WebDriver 是 WebDirver 的子类，它继承了 WebDriver 的所有方法，并在此基础上增加了 wait 方法，用于等待元素出现或消失。

```
WebDriverWait(driver, timeout, poll_frequency=0.5,
ignored_exceptions=None)
```

driver 是 WebDriver 对象

timeout 是等待的超时时间

poll\_frequency 是轮询的频率，默认是 0.5S

ignored\_exceptions 是忽略的异常，包括 NoSuchElementException

WebDriverWait() 的 until() 和 until\_not() 方法，用于等待元素出现或消失。

- until(method, message='')

返回True

· until\_not(method, message= ' ')

返回False

as expected\_conditions EC  
presence\_of\_element\_located()

expected\_conditions4.3

4.3 expected\_conditions

| 名称                               | 说明       |
|----------------------------------|----------|
| title_is                         | 页面标题是    |
| title_contains                   | 页面标题包含   |
| presence_of_element_located      | DOM元素存在  |
| visibility_of_element_located    | 元素可见且不为0 |
| visibility_of                    | 元素可见     |
| presence_of_all_elements_located | 所有元素存在   |



|  |   |
|--|---|
|  | DOM 是否包含<br>具有 n 的 class<br>为 "wp" 的文本<br>是否 True |
| text_to_be_present_in_element          | 元素 text 是否<br>包含                                  |
| text_to_be_present_in_element_value    | 元素 value 是否<br>包含                                 |
| frame_to_be_available_and_switch_to_it | 是否可用并<br>切换到 True<br>是否 switch 是否<br>是否 False     |
| invisibility_of_element_located        | 元素是否<br>DOM 是否                                    |
| element_to_be_clickable                | 元素是否<br>是否  |
| staleness_of                           | DOM 是否<br>是否                                      |
| element_to_be_selected                 | 元素是否<br>是否  |
| element_selection_state_to_be          | 元素是否<br>是否  |
| element_located_selection_state_to_be  | 元素是否<br>元素是否<br>元素是否                              |
| alert_is_present                       | 是否 alert  |

---

expected\_conditions 是 Selenium 中定义的一个类，用于等待页面元素出现或消失。  
is\_displayed() 是 Selenium 中定义的一个方法，用于判断元素是否可见。

## baidu.py

```
from selenium import webdriver
from time import sleep, ctime
driver = webdriver.Firefox()
driver.get("http://www.baidu.com")
print(ctime())
for i in range(10):
    try:
        el = driver.find_element_by_id("kw22")
        if el.is_displayed():
            break
    except: pass
    sleep(1)
else:
    print("time out")
driver.close()
print(ctime())
```

该代码的作用是：在 10 秒内等待百度首页的“kw22”元素出现并可见。  
如果 10 秒内元素出现并可见，则打印“True”并退出循环。  
如果 10 秒内元素未出现或不可见，则打印“time out”并退出循环。

运行结果：

## Python Shell

```
===== RESTART: D:/pyse/baidu.py
=====
Fri Oct 23 22:51:25 2015
time out
Fri Oct 23 22:51:35 2015
```

### 4.7.2 断言

断言（assert）是Python中一个非常强大的工具，用于检查程序中的某些条件是否成立。如果条件不成立，程序就会抛出断言错误（AssertionError）。在WebDriver中，断言可以用于检查页面元素是否存在、是否可见、是否可点击等。例如，在WebDriver中，可以使用断言来检查元素是否存在，如果不存在，就会抛出NoSuchElementException。

baidu.py

```
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from time import ctime
driver = webdriver.Firefox()
# 断言等待10秒
driver.implicitly_wait(10)
driver.get("http://www.baidu.com")
try:
    print(ctime())
    driver.find_element_by_id("kw22").send_keys('selenium')
except NoSuchElementException as e:
```

```
        implicitly_wait(10, 10)
    }

    {
        6, 10
    }

    id=kw22
    10
```

```
===== RESTART: D:/pyse/baidu.py
=====
Fri Oct 23 22:57:05 2015
Message: Unable to locate element:
{"method":"id","selector":"kw22"}
Stacktrace:
    at FirefoxDriver.prototype.findElementInternal_
(file:///C:/Users/fnngj/AppData/Local/Temp/tmpwi0luzkz/extensio
ns/fxdriver@
googlecode.com/components/driver-component.js:10659)
    at fxdriver.Timer.prototype.setTimeout/<.notify
(file:///C:/Users/fnngj/AppData/Local/Temp/tmpwi0luzkz/extensio
ns/fxdriver@
```

googlecode.com/components/driver-component.js:621)

Fri Oct 23 22:57:15 2015

## 4.7.3 sleep

sleep() 函数是 Python 中 time 模块的一个函数

baidu.py

```
from selenium import webdriver
from time import sleep
driver = webdriver.Firefox()
driver.get("http://www.baidu.com")
sleep(2)
driver.find_element_by_id("kw").send_keys("webdriver")
driver.find_element_by_id("su").click()
sleep(3)
driver.quit()
```

sleep() 函数可以接受一个参数，表示睡眠的时间。例如，sleep(0.5) 表示睡眠 0.5 秒。

## 4.8

4.1 Selenium 8 Selenium WebDriver  
8 Selenium

```
find_elements_by_id()

find_elements_by_name()

find_elements_by_class_name()

find_elements_by_tag_name()

find_elements_by_link_text()

find_elements_by_partial_link_text()

find_elements_by_xpath()

find_elements_by_css_selector()
```

Selenium element s  
Selenium

- Selenium
- Selenium checkbox  
Selenium

checkbox.html

```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-
8" />
<title>Checkbox</title>
<link
href="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.
css"
rel="stylesheet" />
<script
src="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.j
s"></script>
</head>
  <body>
    <h3>checkbox</h3>
    <div class="well">
      <form class="form-horizontal">
        <div class="control-group">
          <label class="control-label"
for="c1">checkbox1</label>
          <div class="controls">
            <input type="checkbox" id="c1" />
          </div>
        </div>
        <div class="control-group">
          <label class="control-label"
for="c2">checkbox2</label>
```

```

        <div class="controls">
            <input type="checkbox" id="c2" />
        </div>
    </div>
    <div class="control-group">
        <label class="control-label"
for="c3">checkbox3</label>
        <div class="controls">
            <input type="checkbox" id="c3" />
        </div>
    </div>
</form>
</div>
</body>
</html>

```

000000checkbox.html0000000000000000Bootstrap  
 0000000000000000004.10000

checkbox

checkbox1 ☐

checkbox2 ☐

checkbox3 ☐

4.10 000

000000000000000000000000



## checkbox.py

```
from selenium import webdriver
import os,time
driver = webdriver.Firefox()
file_path = 'file:/// ' + os.path.abspath('checkbox.html')
driver.get(file_path)
# 通过tag name找input
inputs = driver.find_elements_by_tag_name('input')
# 通过type找checkbox
for i in inputs:
    if i.get_attribute('type') == 'checkbox':
        i.click()
        time.sleep(1)
driver.quit()
```

通过tag name找input  
find\_elements\_by\_tag\_name() 通过tag name找input  
for  
get\_attribute() 通过type找checkbox  
“checkbox”

通过html找input  
Python os.path.abspath() 通过path找input

通过XPath CSS找input

## checkbox.py

```

from selenium import webdriver
import os,time
driver = webdriver.Firefox()
file_path = 'file:/// ' + os.path.abspath('checkbox.html')
driver.get(file_path)
# XPath type=checkbox
#
checkboxes =
driver.find_elements_by_xpath("//input[@type='checkbox']")
# CSS type=checkbox
checkboxes =
driver.find_elements_by_css_selector('input[type=checkbox]')
for checkbox in checkboxes:
    checkbox.click()
    time.sleep(1)
# type=checkbox
print(len(checkboxes))
# 1 checkbox
driver.find_elements_by_css_selector('input[type=checkbox]').pop().click()
driver.quit()

```

XPath CSS

Python len()
 print() pop()

pop().lick() 调用 pop() 方法  
将栈顶元素弹出

pop() 弹出栈顶元素  
pop(-1) 弹出栈顶元素  
pop(0) 弹出栈顶元素  
pop(1) 弹出栈顶元素  
.....

将栈顶元素弹出

## 4.9 帧

Web 页面中的 frame/iframe 元素由 WebDriver 驱动  
frame/iframe 元素由 switch\_to.frame() 方法  
switch\_to.frame() 方法

frame.html

```
<html>
<head>
<link
href="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.
css"
rel="stylesheet" />
<script
type="text/javascript">$(document).ready(function(){
    });
</script>
```

```

</head>
<body>
<div class="row-fluid">
    <div class="span10 well">
        <h3>frame
                                <iframe    id="if"    name="nf"
src="http://www.baidu.com" width="800"
                                height="300">
        </iframe>
    </div>
</div>
</body>
<script
src="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.js"></script>
</html>

```

HTML 使用 iframe 嵌入外部页面 4.11



4.11 iframe 示例

```
switch_to.frame()frame.html<iframe>

```

frame.py

```
from selenium import webdriver
import time
import os
driver = webdriver.Firefox()
file_path = 'file:/// ' + os.path.abspath('frame.html')
driver.get(file_path)
# iframeid = "if"
driver.switch_to.frame("if")
#
driver.find_element_by_id("kw").send_keys("selenium")
driver.find_element_by_id("su").click()
time.sleep(3)
driver.quit()
```

```
switch_to.frame()idnameiframe
idname
```

frame.py

```
.....
# xpathiframe
xf = driver.find_element_by_xpath('//*[class="if"]')
```

```
# 切换到父级frame
driver.switch_to.frame(xf)

.....

driver.switch_to.parent_frame()
```

```
切换到父级content
driver.switch_to.frame(xf)
driver.switch_to.default_content()
```

## 4.10 窗口

WebDriver 提供了 `switch_to.window()` 方法，用于切换到指定的窗口。

4.12 窗口



4.12 窗口

## windows.py

```
from selenium import webdriver
import time
driver = webdriver.Firefox()
driver.implicitly_wait(10)
driver.get("http://www.baidu.com")
# 打开百度
sreach_windows = driver.current_window_handle
driver.find_element_by_link_text('百度').click()
driver.find_element_by_link_text("百度").click()
# 打开百度搜索
all_handles = driver.window_handles
# 遍历所有窗口
for handle in all_handles:
    if handle != sreach_windows:
        driver.switch_to.window(handle)
        print('now register window!')

driver.find_element_by_name("account").send_keys('username')

driver.find_element_by_name('password').send_keys('password')
time.sleep(2)
# .....
# 关闭所有窗口
for handle in all_handles:
    if handle == sreach_windows:
```

```
driver.switch_to.window(handle)
print('now sreach window!')
```

```
driver.find_element_by_id('TANGRAM__PSP_2__closeBtn').click()
driver.find_element_by_id("kw").send_keys("selenium")
driver.find_element_by_id("su").click()
time.sleep(2)
driver.quit()
```

```
current_window_handle
sreach_handle
window_handles
all_handles
```

```
all_handles
handle
sreach_handle
switch_to.window()
handle
sreach_handle
```

```
current_window_handle
```

```
window_handles
```

```
switch_to.window()
switch_to.frame()

```



## 4.11 弹窗

WebDriver 的 JavaScript 的 alert、confirm、prompt 方法，通过 switch\_to\_alert() 切换到 alert/confirm/prompt 窗口，通过 text/accept/dismiss/ send\_keys 操作。

- text 切换到 alert/confirm/prompt 窗口
- accept() 接受
- dismiss() 拒绝
- send\_keys(keysToSend) 向 keysToSend 输入文本

图 4.13 展示了 WebDriver 的 switch\_to\_alert() 方法。

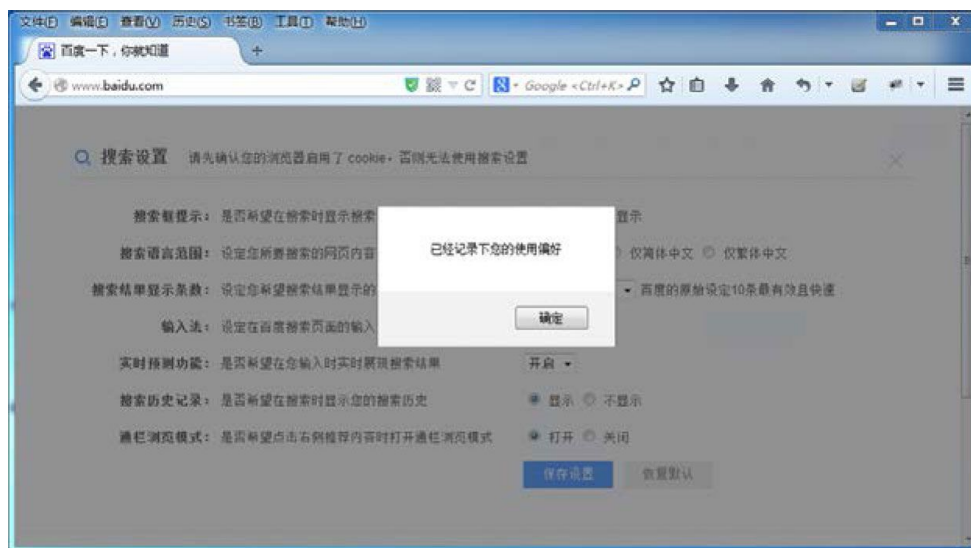


图 4.13 弹窗

alert\_.py

```

from selenium import webdriver
from selenium.webdriver.common.action_chains import
ActionChains
import time
driver = webdriver.Firefox()
driver.implicitly_wait(10)
driver.get('http://www.baidu.com')
# 定位“百度”
link = driver.find_element_by_link_text('百度')
ActionChains(driver).move_to_element(link).perform()
# 单击
driver.find_element_by_link_text("百度").click()
# 悬停
driver.find_element_by_class_name("prefpanelgo").click()
time.sleep(2)
# 处理弹窗
driver.switch_to_alert().accept()
driver.quit()

```

通过本章的学习，我们了解了 ActionChains 类，通过 move\_to\_element() 方法可以定位元素并移动到元素上方，通过 click() 方法可以单击元素，通过 switch\_to\_alert() 方法可以切换到弹出的 alert 对话框，通过 accept() 方法可以接受 alert 对话框。

## 4.12 小结

WebDriver 的 WebDriver 接口，WebDriver 接口是 WebDriver 的基接口，所有 WebDriver 的实现都必须实现这个接口。

WebDriver 接口定义了一个 `getWindow()` 方法，这个方法返回一个 `Window` 接口，这个接口是 `WebDriver` 接口的一部分，它定义了 `Window` 接口的方法。

WebDriver 接口定义了一个 `getWindow()` 方法，这个方法返回一个 `Window` 接口，这个接口是 `WebDriver` 接口的一部分，它定义了 `Window` 接口的方法。

- 这个接口定义了一个 `input` 方法，这个方法返回一个 `form` 接口，这个接口是 `Window` 接口的一部分，它定义了 `form` 接口的方法。
- 这个接口定义了一个 `Flash` 方法，这个方法返回一个 `JavaScript` 接口，这个接口是 `Window` 接口的一部分，它定义了 `JavaScript` 接口的方法。

## 4.12.1 send\_keys

这个接口定义了一个 `send_keys()` 方法，这个方法返回一个 `sendKeys()` 接口，这个接口是 `Window` 接口的一部分，它定义了 `sendKeys()` 接口的方法。

upfile.html

```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>upload_file</title>
<link
href="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.
css"
```

```

rel="stylesheet" />
</head>
<body>
    <div class="row-fluid">
        <div class="span6 well">
            <h3>upload_file</h3>
            <input type="file" name="file" />
        </div>
    </div>
</body>
<script
src="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.j
s"></script>
</html>

```

upfile.html 4.14



4.14

upfile.py

```

from selenium import webdriver
import os
driver = webdriver.Firefox()
file_path = 'file:/// ' + os.path.abspath('upfile.html')

```

```
driver.get(file_path)
# 0000000000000000
driver.find_element_by_name("file").send_keys('D:\\upload_file.
txt')
driver.quit()
```

0000000000000000Windows000000000000input000000  
000000send\_keys()00000000000000000000

## 4.12.2 Autolt0000

Autolt000000v3000000000BASIC0000000000000000  
Windows GUI(000000)000000000000000000000000/00000000  
000000

00000<https://www.autoitscript.com/site/>

000000Autolt00000000000000000000004.15000Auto Lt000

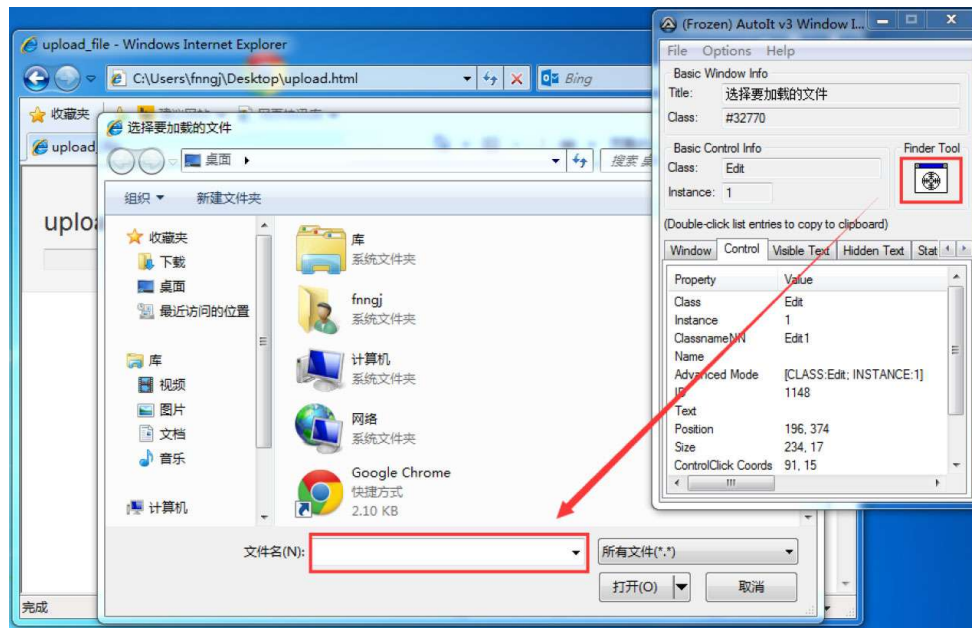
Autolt Windows Info 0000Windows00000  
Compile Script to.exe 000Autolt00exe00000  
Run Script 0000Autolt000  
SciTE Script Editor 0000Autolt000



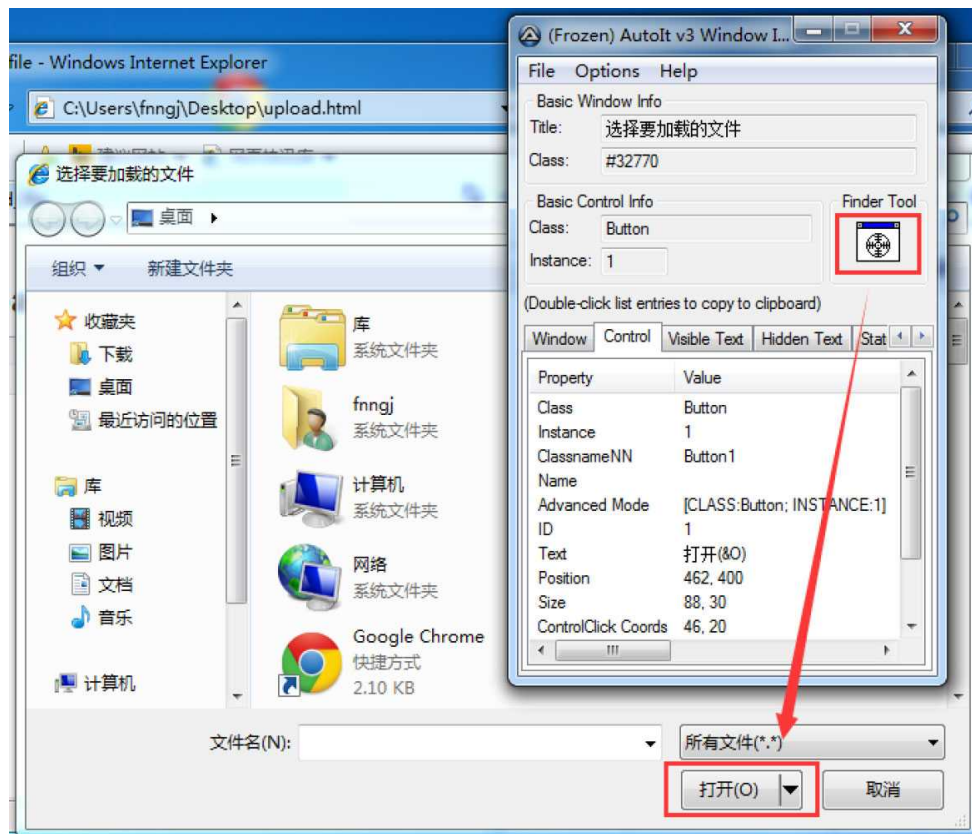
图4.15 AutoIt的启动菜单

在<http://www.autoit.com.au/upload.html>上可以下载到AutoIt的源代码

1. 在AutoIt Windows Info窗口中单击Finder Tool按钮，如图4.16所示。在弹出的对话框中单击“是”按钮，如图4.17所示。



4.16 AutoIt Windows Info “ ”



4.17 AutoIt Windows Info “ ”

4.16 4.17 Autolt Windows Info

title " " Class "#32770"

class "Edit" Instance "1" ClassnameNN  
"Edit1"

class "Button" Instance "1" ClassnameNN  
"Button1"

2 Autolt Windows Info SciTE Script  
Editor Autolt

upfile.au3

```
;ControlFocus("title","text",controlID) Edit1=Edit instance 1
ControlFocus("","Edit1")
; Wait 10 seconds for the Upload window to appear
WinWait("[CLASS:#32770]", "",10)
; Set the File name text on the Edit field
ControlSetText(" ", "", "Edit1",
"D:\\upload_file.txt")
Sleep(2000)
; Click on the Open button
ControlClick("","Button1");
```

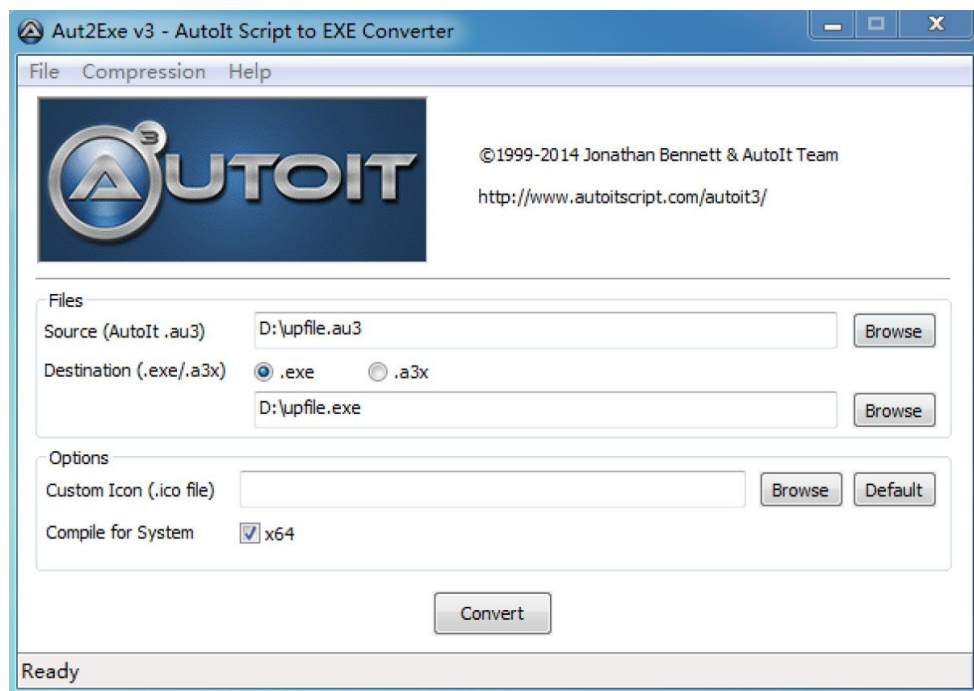
ControlFocus() Window WinWait() 10  
ControlSetText() " " Sleep()  
Sleep() Python time Sleep()



Sleep(2000) 2000 ControlClick() “”

AutoIt “Tools”→“Go” F5

3 upfile.au3 Run Script  
Python exe  
Compile Script to.exe exe 4.18



4.18 Compile Script to.exe exe

“Browse” upfile.au3 “Convert” upfile.exe

4. Selenium WebDriver를 사용하여 upfile.exe 실행하기

upfile.py

```
from selenium import webdriver
import os
driver = webdriver.Firefox()
# Selenium WebDriver
file_path = 'file:/// ' + os.path.abspath('upfile.html')
driver.get(file_path)
# Selenium WebDriver
driver.find_element_by_name("file").click()
# upfile.exe 실행
os.system("D:\\upfile.exe")
driver.quit()
```

os.system()을 사용하여 upfile.exe 실행

WebDriver를 사용하여 upfile.html을 로드하고, 'file' 이름의 요소에 클릭하여 upfile.exe를 실행합니다. Python에서 os.system()을 사용하여 upfile.exe를 실행하는 방법도 있습니다.

## 4.13 Selenium WebDriver를 사용하여 downfile.exe 실행하기

WebDriver를 사용하여 downfile.html을 로드하고, 'file' 이름의 요소에 클릭하여 downfile.exe를 실행합니다.

downfile.py

```

from selenium import webdriver
import os
fp = webdriver.FirefoxProfile()
fp.set_preference("browser.download.folderList",2)
fp.set_preference("browser.download.manager.showWhenStarting",False)
fp.set_preference("browser.download.dir", os.getcwd())
fp.set_preference("browser.helperApps.neverAsk.saveToDisk",
"application/octet-stream") #application/octet-stream
driver = webdriver.Firefox(firefox_profile=fp)
driver.get("http://pypi.Python.org/pypi/selenium")
driver.find_element_by_partial_link_text("selenium-2").click()

```

FirefoxFirefoxProfile()

browser.download.folderList

02

browser.download.manager.showWhenStarting

TrueFalse

browser.download.dir

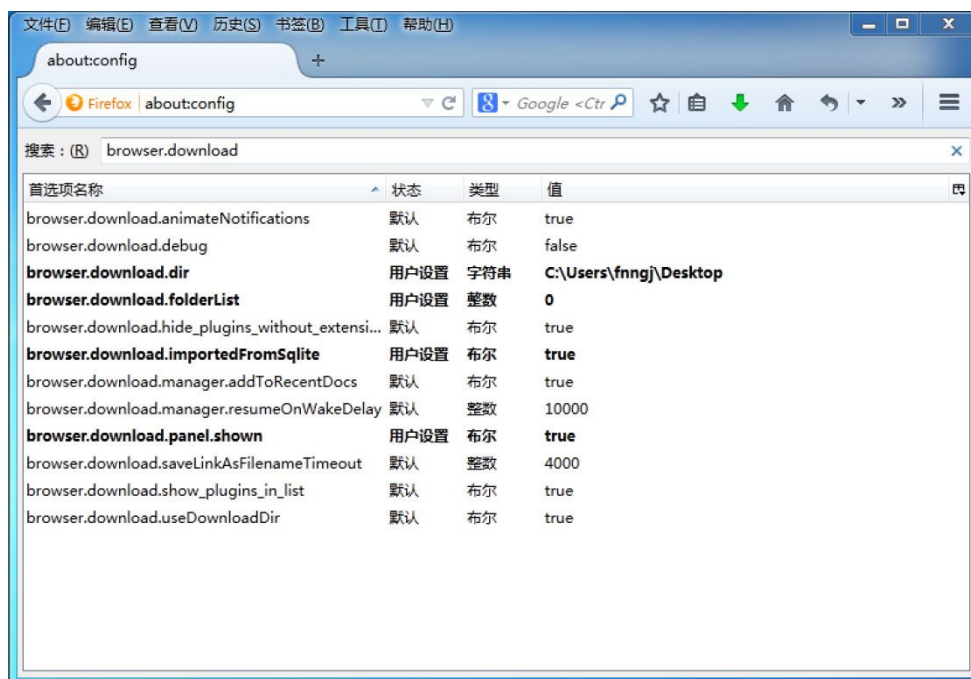
os.getcwd()

browser.helperApps.neverAsk.saveToDisk

Content-type“application/octet-stream”

HTTP Content-type http://tool.oschina.net/commons

Firefoxabout:config4.19



4.19 Firefox

WebDriverFirefox()Firefox

FirefoxAutoltWindows4.13Autolt

## 4.14 Cookie

WebDriver 提供了 cookie 接口，通过 `WebDriver` 接口可以获取、设置、删除 cookie。

WebDriver 提供了 `cookie` 接口。

- `get_cookies()` 获取所有 cookie。
- `get_cookie(name)` 根据 key “name” 获取 cookie。
- `add_cookie(cookie_dict)` 设置 cookie “cookie\_dict”，其中 `cookie_dict` 是一个字典，键为 name，值为 value。
- `delete_cookie(name,optionsString)` 删除 cookie “name”，其中 `optionsString` 是一个字符串，格式为 “name” “value”。
- `delete_all_cookies()` 删除所有 cookie。

下面通过 `get_cookies()` 获取所有 cookie。

cookie.py

```
from selenium import webdriver
driver = webdriver.Firefox()
driver.get("http://www.youdao.com")
# 获取 cookie
cookie= driver.get_cookies()
# 打印 cookie
print(cookie)
driver.quit()
```

□□□□

===== RESTART: D:/pyse/cookie.py

=====

```
[{'httpOnly': False, 'name': 'YOUDAO_MOBILE_ACCESS_TYPE',  
'secure': False,  
'expiry': 1477324093, 'value': '1', 'path': '/', 'domain':  
'youdao.com'},
```

```
{'httpOnly': False, 'name': 'OUTFOX_SEARCH_USER_ID', 'secure':  
False, 'expiry':  
2391868093, 'value': '2045063256@112.90.37.235', 'path': '/',  
'domain':
```

```
'youdao.com'}, {'httpOnly': False, 'name': 'JSESSIONID',  
'secure': False,  
'expiry': None, 'value': 'abcWVA6WwVSR2bIW4RFcv', 'path': '/',  
'domain':
```

```
'www.youdao.com'}, {'httpOnly': False, 'name':  
'CNZZDATA1256118513', 'secure':  
False, 'expiry': 1461512894, 'value': '1183881050-1445787498-  
%7C1445787498',
```

```
'path': '/', 'domain': 'www.youdao.com'}, {'httpOnly': False,  
'name':
```

```
'lzstat_uv', 'secure': False, 'expiry': 1761407294, 'value':  
'13013917332303058731|3601912', 'path': '/', 'domain':  
'youdao.com'},
```

```
{'httpOnly': False, 'name': 'lzstat_ss', 'secure': False,  
'expiry': None,  
'value': '449876553_0_1445816894_3601912', 'path': '/',
```

```
'domain':  
'youdao.com']}]
```

```
cookiecookiecookie  
cookie
```

## cookie.py

```
from selenium import webdriver  
driver = webdriver.Firefox()  
driver.get("http://www.youdao.com")  
# cookie name value  
driver.add_cookie({'name': 'key-aaaaaaa', 'value': 'value-  
bbbbbb'})  
# cookies name value  
for cookie in driver.get_cookies():  
    print("%s -> %s" % (cookie['name'], cookie['value']))  
driver.quit()  
  
===== RESTART: D:/pyse/cookie.py  
=====
```

YOUDAO\_MOBILE\_ACCESS\_TYPE -> 1  
\_PREF\_ANONYUSER\_\_MYTH -> aGFzbG9nZ2VkPXRydWU=  
OUTFOX\_SEARCH\_USER\_ID -> -1046383847@218.17.158.115  
JSESSIONID -> abc7qSE\_SBGsVgnVLBvcu  
key-aaaaaaa -> value-bbbbbbb

WebDriver.add\_cookie(cookie\_dict) 方法添加 cookie。cookie\_dict 字典的 key 为 "name" 和 "value"，value 为 cookie 的值。

WebDriver.get\_cookies() 方法返回所有 cookie。返回的字典的 key 为 "username"，value 为 username。字典的 key 为 "value"，value 为 cookie 的值。

WebDriver.delete\_cookie(cookie\_dict) 方法删除 cookie。cookie\_dict 字典的 key 为 "name"，value 为 cookie 的值。

## 4.15 JavaScript

WebDriver.execute\_script() 方法执行 JavaScript 脚本。脚本的 key 为 "script"，value 为 JavaScript 脚本。

WebDriver.execute\_script() 方法执行 JavaScript 脚本。脚本的 key 为 "script"，value 为 JavaScript 脚本。

WebDriver.execute\_script("JavaScript")

html

.....

<!-- window.scrollTo(0,0); -->



```
window.scrollTo(0,450);
```

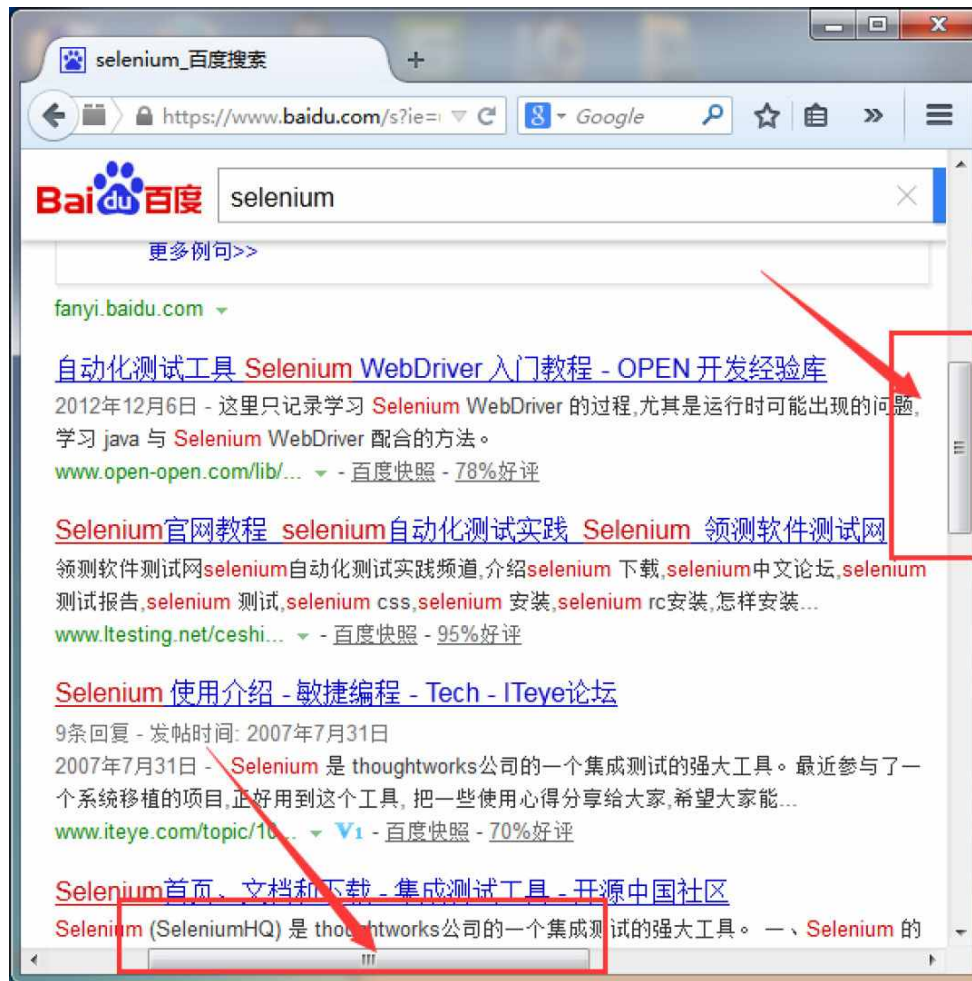
.....

```
    window.scrollTop()
    浏览器窗口滚动到顶部
```

## baidu.py

```
from selenium import webdriver
from time import sleep
# 浏览器
driver=webdriver.Firefox()
driver.get("http://www.baidu.com")
# 窗口大小
driver.set_window_size(600, 600)
# 搜索
driver.find_element_by_id("kw").send_keys("selenium")
driver.find_element_by_id("su").click()
sleep(2)
# 执行JavaScript
js="window.scrollTo(100,450);"
driver.execute_script(js)
sleep(3)
driver.quit()
```

```
    设置窗口大小
    执行JavaScript
    JavaScript
    4.20
```



4.20 JavaScript

JavaScript  
4.21

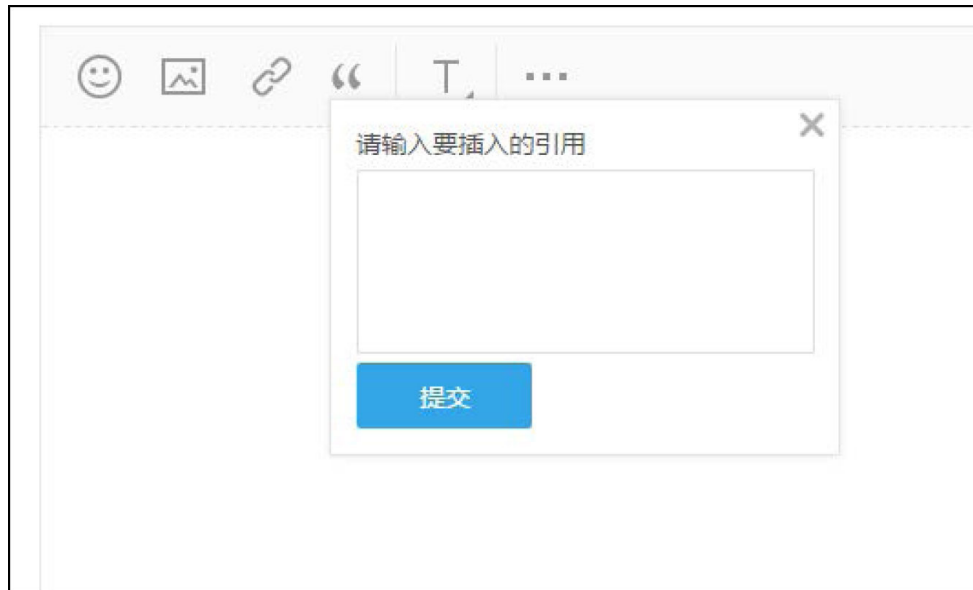


图4.21 对话框

下面是一个例子

html

.....

```
<textarea id="id" style="width: 98%" cols="50" rows="5"
class="txtarea">
</textarea>
```

.....

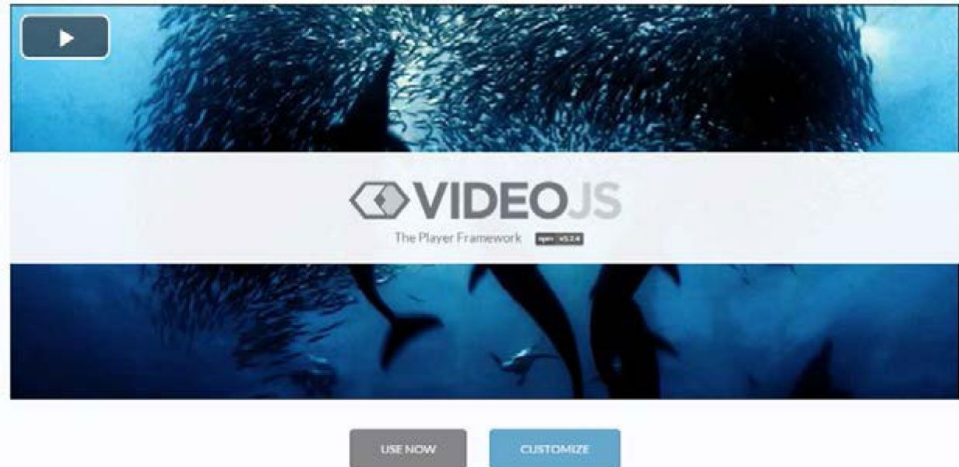
下面是一个例子，它使用id属性来调用JavaScript的send\_keys()方法，该方法可以向文本框中输入文本。

js\_test.py

.....

```
text = "input text"
```





## 4.22 HTML5 Video Player

このセクションでは、HTML5の<video>タグとJavaScriptを使用して、  
ビデオプレイヤーを構築する方法を説明します。

### test\_video.py

```
from selenium import webdriver
from time import sleep

driver = webdriver.Firefox()
driver.get("http://videojs.com/")

video = driver.find_element_by_xpath("body/Section[1]/div/video")

# URLを取得
url = driver.execute_script("return arguments[0].currentSrc;",
video)
print(url)

# 再生
print("start")
driver.execute_script("return arguments[0].play()", video)
```

```
# 等待15秒
sleep(15)
# 打印
print("stop")
driver.execute_script("arguments[0].pause()", video)
driver.quit()
```

JavaScript 的 `arguments` 对象是一个类数组对象，它的长度属性 `arguments.length` 返回该对象的长度。

`currentSrc` 属性返回当前正在播放的视频的 URL。

`load()`、`play()`、`pause()` 方法分别用于加载、播放和暂停视频。

## 4.17 截图

在本节中，我们将使用 Selenium 的 `WebDriver` 接口来截取网页的截图。我们使用 `get_screenshot_as_file()` 方法来截取网页的截图。

`baidu.py`

```
from selenium import webdriver
from time import sleep
driver = webdriver.Firefox()
driver.get('http://www.baidu.com')
driver.find_element_by_id('kw').send_keys('selenium')
driver.find_element_by_id('su').click()
```

```
sleep(2)
# 保存图片到本地
driver.get_screenshot_as_file("D:\\pyse\\baidu_img.jpg")
driver.quit()
```

保存图片到本地D:\\pyse\\baidu\_error.jpg

## 4.18 保存图片

调用`quit()`方法关闭浏览器窗口，同时也会关闭`WebDriver`对象。`close()`方法只关闭当前窗口。4.10版本开始，`close()`方法也可以用于关闭整个浏览器窗口。

## 4.19 关闭浏览器

调用`Web`对象的`quit()`方法可以关闭浏览器窗口。调用`close()`方法可以关闭当前窗口。调用`quit()`方法可以关闭整个浏览器窗口。

调用`Web`对象的`quit()`方法可以关闭浏览器窗口。调用`close()`方法可以关闭当前窗口。调用`quit()`方法可以关闭整个浏览器窗口。

## 1 保存图片





randint()ランダムに1000~9999の整数を生成する  
プログラムを作成してください

## Python Shell

===== RESTART: D:/pyse/number.py

=====

ランダム: 8396

ランダム: 8396

8396

ランダム!!

===== RESTART: D:/pyse/number.py

=====

ランダム: 5113

ランダム: 132741

132741

ランダム!!

===== RESTART: D:/pyse/number.py

=====

ランダム: 1996

ランダム: 1234

1234

ランダム

**3**ランダム

Python-tesseract Python-tesseract  
Tesseract OCR Python (JPG GIF  
PNG TIFF) 100%

## 4 cookie

cookie  
“”  
“” cookie WebDriver  
cookie add\_cookie() cookie  
cookie

cookie\_login.py

```
# .....  
# xx  
driver.get("http://www.xx.cn")  
# cookie  
driver.add_cookie({'name': 'Login_UserNumber',  
                  'value': 'username'})  
driver.add_cookie({'name': 'Login_Passwd', 'value': 'password'})  
# xx  
driver.get("http://www.xx.cn/")  
# .....  
driver.quit()
```

cookie key  
get\_cookies() cookie key

## 4.20 WebDriver

WebDriver Server - Client

Server Remote Server  
Remote Server Client

Client  
URL http Remote Server Remote  
Server Response

WebDriver

① WebDriver  
WebDriver Remote Server

② Client CommandExcuter HTTPRequest Remote  
Server the webriver wire protocol

③ Remote Server IEDriverServer.exe  
chromedriver.exe native

Python logging  
basicConfig() debug

## baidu.py

```
from selenium import webdriver
import logging
logging.basicConfig(level=logging.DEBUG)
diver = webdriver.Firefox()
diver.get("http://www.baidu.com")
diver.find_element_by_id("kw").send_keys("selenium")
diver.find_element_by_id("su").click()
diver.quit()
```

```
basicConfig()
log
basicConfig()
debug
POST
Selenium Server
Selenium Server
```

## Python Shell

```
===== RESTART: D:/pyse/baidu.py
=====
DEBUG:selenium.webdriver.remote.remote_connection:POST
http://127.0.0.1:34229/hub/session {"desiredCapabilities":
{"platform": "ANY",
"browserName": "firefox", "version": "", "javascriptEnabled":
true}}
DEBUG:selenium.webdriver.remote.remote_connection:Finished
Request
DEBUG:selenium.webdriver.remote.remote_connection:POST
http://127.0.0.1:34229/hub/session/0f0d51f5-affc-4af0-9c45-
```

4b3c4931c601/url

```
{"url": "http://www.baidu.com", "sessionId":  
"0f0d51f5-affc-4af0-9c45-4b3c4931c601"}
```

DEBUG:selenium.webdriver.remote.remote\_connection:Finished

Request

DEBUG:selenium.webdriver.remote.remote\_connection:POST

http://127.0.0.1:34229/hub/session/0f0d51f5-affc-4af0-9c45-  
4b3c4931c601/ele

```
ment {"using": "id", "sessionId": "0f0d51f5-affc-4af0-9c45-  
4b3c4931c601",  
"value": "kw"}
```

DEBUG:selenium.webdriver.remote.remote\_connection:Finished

Request

DEBUG:selenium.webdriver.remote.remote\_connection:POST

http://127.0.0.1:34229/hub/session/0f0d51f5-affc-4af0-9c45-  
4b3c4931c601/ele

```
ment/{12722a5d-58f3-457c-ad5e-348b230c6f6a}/value {"sessionId":  
"0f0d51f5-affc-4af0-9c45-4b3c4931c601", "id":  
"{12722a5d-58f3-457c-ad5e-348b230c6f6a}", "value": ["s", "e",  
"l", "e", "n",  
"i", "u", "m"]}
```

DEBUG:selenium.webdriver.remote.remote\_connection:Finished

Request

DEBUG:selenium.webdriver.remote.remote\_connection:POST

http://127.0.0.1:34229/hub/session/0f0d51f5-affc-4af0-9c45-  
4b3c4931c601/ele

```
ment {"using": "id", "sessionId": "0f0d51f5-affc-4af0-9c45-
```

```
4b3c4931c601",
  "value": "su"}
DEBUG:selenium.webdriver.remote.remote_connection:Finished
Request
DEBUG:selenium.webdriver.remote.remote_connection:POST
http://127.0.0.1:34229/hub/session/0f0d51f5-affc-4af0-9c45-
4b3c4931c601/element/{8090ac84-2d92-4b48-8320-dadcfbf15f40}/click {"sessionId":
"0f0d51f5-affc-4af0-9c45-4b3c4931c601", "id":
"{8090ac84-2d92-4b48-8320-dadcfbf15f40}"}
DEBUG:selenium.webdriver.remote.remote_connection:Finished
Request
DEBUG:selenium.webdriver.remote.remote_connection:DELETE
http://127.0.0.1:34229/hub/session/0f0d51f5-affc-4af0-9c45-
4b3c4931c601
{"sessionId": "0f0d51f5-affc-4af0-9c45-4b3c4931c601"}
DEBUG:selenium.webdriver.remote.remote_connection:Finished
Request
```



```

0000210000000000000000000000WebDriver00000Web
0000000000000000

```

[illegible]

本書では、Selenium WebDriverのインストールと実行方法、そして、WebDriverのAPIの概要について説明します。

本書では、UIの自動化を行うためのWebDriverのインストールと実行方法、そして、WebDriverのAPIの概要について説明します。

本書では、XPath、CSS、JavaScript、jQuery、Webdriverのインストールと実行方法、そして、WebDriverのAPIの概要について説明します。

本書では、Webdriverのインストールと実行方法、そして、WebDriverのAPIの概要について説明します。

本書では、WebDriverのインストールと実行方法、そして、WebDriverのAPIの概要について説明します。

本書では、WebDriverのインストールと実行方法、そして、WebDriverのAPIの概要について説明します。

本書では、WebDriverのインストールと実行方法、そして、WebDriverのAPIの概要について説明します。

1 XPath\CSSのインストールと実行方法、そして、WebDriverのAPIの概要について説明します。

2 WebDriver APIのインストールと実行方法、そして、WebDriverのAPIの概要について説明します。

3 JavaScript\jQueryのインストールと実行方法、そして、WebDriverのAPIの概要について説明します。





## 5 测试框架

---

测试框架是指用于组织测试代码、管理测试用例、执行测试并报告结果的软件工具。

测试框架通常包含以下组件：  
1. **Library**：测试用例的集合。  
2. **WebDriver**：用于控制浏览器的接口。  
3. **Web**：被测应用。  
4. **Web**：被测应用。

测试框架通常包含以下组件：  
1. **Framework**：测试用例的组织结构。  
2. **unittest**：Python 的单元测试框架。  
3. **TestRunner**：用于执行测试并报告结果的类。  
4. **TestResult**：用于存储测试结果的对象。

测试框架通常包含以下组件：  
1. **Tools**：用于测试的工具。  
2. **Selenium IDE**：用于记录和执行测试用例的工具。  
3. **QTP**：用于测试的工具。

测试框架通常包含以下组件：  
1. **TestRunner**：用于执行测试并报告结果的类。  
2. **TestResult**：用于存储测试结果的对象。

### 5.1 测试框架

测试框架是指用于组织测试代码、管理测试用例、执行测试并报告结果的软件工具。

#### 5.1.1 测试

脚本01:
 from selenium import webdriver
 import time

 driver = webdriver.Firefox()
 driver.get("http://www.xxx.com")
 driver.find\_element\_by\_id("xxx")
 .....

 脚本02:
 from selenium import webdriver
 import time

 driver = webdriver.Firefox()
 driver.get("http://www.xxx.com")
 driver.find\_element\_by\_id("xxx")
 .....

 脚本03:
 from selenium import webdriver
 import time

 driver = webdriver.Firefox()
 driver.get("http://www.xxx.com")
 driver.find\_element\_by\_id("xxx")
 .....

图5.1 脚本01

脚本01:
 from selenium import webdriver
 import time

 driver = webdriver.Firefox()
 driver.get("http://www.xxx.com")
 driver.find\_element\_by\_id("xxx")
 .....

- 脚本02:
 from selenium import webdriver
 import time

 driver = webdriver.Firefox()
 driver.get("http://www.xxx.com")
 driver.find\_element\_by\_id("xxx")
 .....
- 脚本03:
 from selenium import webdriver
 import time

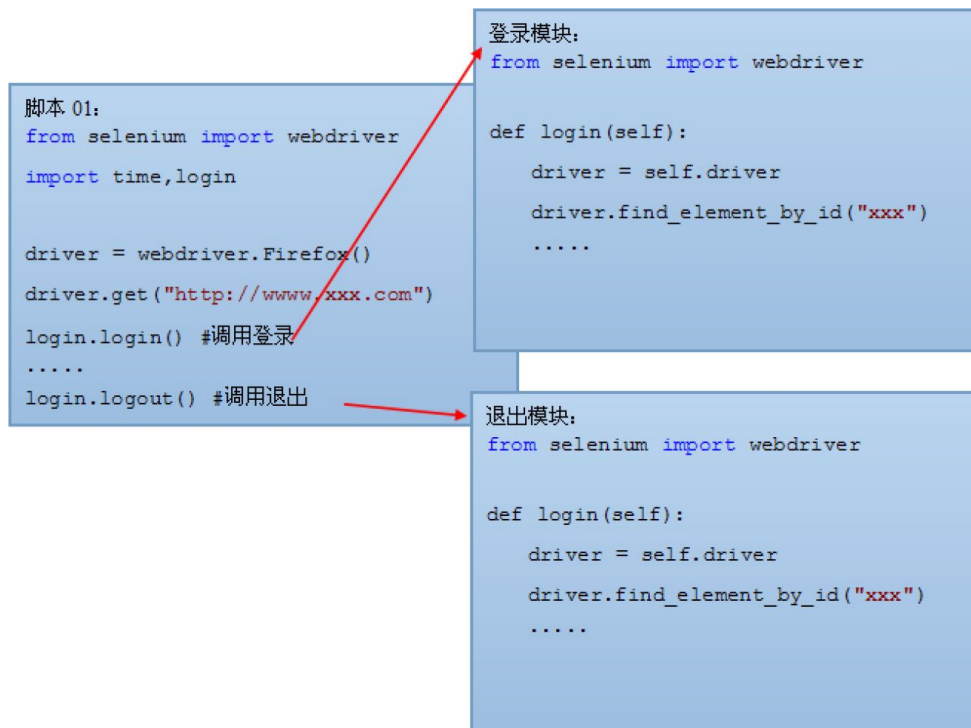
 driver = webdriver.Firefox()
 driver.get("http://www.xxx.com")
 driver.find\_element\_by\_id("xxx")
 .....

## 5.1.2 模块的调用

模块的调用是指在一个模块中调用另一个模块中的函数或类。在Python中，模块的调用是通过import语句实现的。import语句有两种基本形式：import 模块名和from 模块名 import 子模块名。前者将整个模块导入，后者只导入模块中的特定子模块。

在5.2节中，我们将看到如何调用模块中的函数。在5.2节中，我们将看到如何调用模块中的函数。

- 模块的调用是指在一个模块中调用另一个模块中的函数或类。在Python中，模块的调用是通过import语句实现的。import语句有两种基本形式：import 模块名和from 模块名 import 子模块名。前者将整个模块导入，后者只导入模块中的特定子模块。
- 模块的调用是指在一个模块中调用另一个模块中的函数或类。在Python中，模块的调用是通过import语句实现的。import语句有两种基本形式：import 模块名和from 模块名 import 子模块名。前者将整个模块导入，后者只导入模块中的特定子模块。

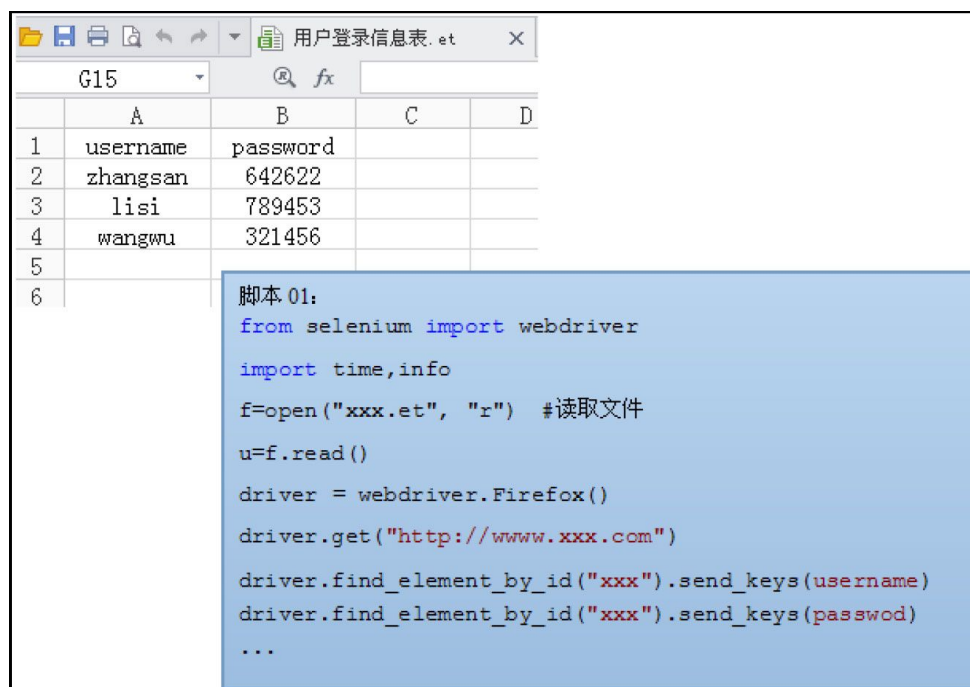


## 5.1.3 数据源

数据源是指数据从哪里来，数据源可以是数据库、文件、接口、爬虫等。在测试过程中，数据源的选择会影响测试的效率和准确性。例如，从数据库中读取数据可以保证数据的完整性和一致性，而从文件中读取数据则更加灵活和方便。

在测试过程中，数据源的选择会影响测试的效率和准确性。例如，从数据库中读取数据可以保证数据的完整性和一致性，而从文件中读取数据则更加灵活和方便。Datapool是一个数据源管理工具，可以帮助用户管理多个数据源，并方便地在测试用例中使用。

5.3 数据源管理



The screenshot shows a web browser window with a table titled "用户登录信息表.et". The table has columns A, B, C, and D. The data rows are as follows:

|   | A        | B        | C | D |
|---|----------|----------|---|---|
| 1 | username | password |   |   |
| 2 | zhangsan | 642622   |   |   |
| 3 | lisi     | 789453   |   |   |
| 4 | wangwu   | 321456   |   |   |
| 5 |          |          |   |   |
| 6 |          |          |   |   |

Below the table, there is a code editor window titled "脚本 01:" containing the following Selenium WebDriver script:

```
脚本 01:
from selenium import webdriver
import time,info
f=open("xxx.et", "r") #读取文件
u=f.read()
driver = webdriver.Firefox()
driver.get("http://www.xxx.com")
driver.find_element_by_id("xxx").send_keys(username)
driver.find_element_by_id("xxx").send_keys(password)
...
```

5.3 数据源管理

数据源管理是指对数据源进行配置、维护和监控的过程。在测试过程中，数据源的管理非常重要，因为它直接影响到测试的效率和准确性。通过合理的数据源管理，可以确保测试数据的完整性和一致性，从而提高测试的质量和效率。

测试用例的编写是测试工作的核心，也是测试人员必须具备的基本技能。测试用例的编写应该遵循一定的原则，包括：测试用例应该覆盖所有的测试需求；测试用例应该具有可执行性；测试用例应该具有可重复性；测试用例应该具有可维护性。在编写测试用例时，应该使用清晰、简洁的语言，避免使用模糊、歧义的语言。同时，还应该对测试用例进行编号，以便于管理和追踪。

## 5.1.4 测试用例的编写

测试用例的编写应该遵循一定的原则，包括：测试用例应该覆盖所有的测试需求；测试用例应该具有可执行性；测试用例应该具有可重复性；测试用例应该具有可维护性。

测试用例的编写应该遵循一定的原则，包括：测试用例应该覆盖所有的测试需求；测试用例应该具有可执行性；测试用例应该具有可重复性；测试用例应该具有可维护性。在编写测试用例时，应该使用清晰、简洁的语言，避免使用模糊、歧义的语言。同时，还应该对测试用例进行编号，以便于管理和追踪。

测试用例的编写应该遵循一定的原则，包括：测试用例应该覆盖所有的测试需求；测试用例应该具有可执行性；测试用例应该具有可重复性；测试用例应该具有可维护性。

| Baidu Test Case (Selenium IDE) |                      |          |
|--------------------------------|----------------------|----------|
| open                           | http://www.baidu.com |          |
| type                           | id=kw                | selenium |
| click                          | id=su                |          |

测试用例的编写应该遵循一定的原则，包括：测试用例应该覆盖所有的测试需求；测试用例应该具有可执行性；测试用例应该具有可重复性；测试用例应该具有可维护性。

- 测试用例应该覆盖所有的测试需求
- 测试用例应该具有可执行性

- 测试用例中测试用例名称“selenium”

测试用例中测试用例名称“selenium”Robot Framework

## 1 if

| If (Robot Framework) |              |          |            |           |
|----------------------|--------------|----------|------------|-----------|
| \${a}                | Set variable | 2        |            |           |
| \${b}                | Set variable | 5        |            |           |
| run keyword if       | \${a}>=1     | log      | a 大于 1     |           |
| ...                  | ELSE IF      | \${b}<=5 | log        | b 小于或等于 5 |
| ...                  | ELSE         | log      | 上面两个条件都不满足 |           |

测试用例中a b 测试用例名称2 5测试用例名称run keyword if测试用例名称a 测试用例名称1测试用例名称log“a 测试用例名称1”测试用例名称b 测试用例名称5测试用例名称log“b 测试用例名称5”测试用例名称log“测试用例名称”

## 2 for

| For (Robot Framework) |       |          |    |
|-----------------------|-------|----------|----|
| :FOR                  | \${i} | in range | 10 |
|                       | log   | \${i}    |    |

测试用例名称for测试用例名称i 1 10测试用例名称log测试用例名称i

## 3

| ReadFile (Robot Framework) |                                       |
|----------------------------|---------------------------------------|
| Import Resource            | \${CURDIR}/resource.txt               |
| Import Resource            | \${CURDIR}/../resources/resource.html |

Import Resource

## 4 import

| Import (Robot Framework) |                             |      |           |         |
|--------------------------|-----------------------------|------|-----------|---------|
| Import Library           | MyLibrary                   |      |           |         |
| Import Library           | \${CURDIR}/Libaray.py       | esom | args      |         |
| Import Library           | \${CURDIR}/../libs/Lib.java | arg  | WITH NAME | JavaLib |

Import Library

## 5.2

Python爬虫入门之爬取126邮箱

mail126.py

```
from selenium import webdriver
driver = webdriver.Firefox()
driver.implicitly_wait(10)
driver.get("http://www.126.com")
# 清除
driver.find_element_by_id("idInput").clear()
driver.find_element_by_id("idInput").send_keys("username")
driver.find_element_by_id("pwdInput").clear()
driver.find_element_by_id("pwdInput").send_keys("password")
driver.find_element_by_id("loginBtn").click()
# 发送邮件
# .....
# 退出
driver.find_element_by_link_text("退出").click()
driver.quit()
```

126邮箱发送邮件

发送邮件



```
from selenium import webdriver

# 浏览器驱动
def login():
    driver.find_element_by_id("idInput").clear()
    driver.find_element_by_id("idInput").send_keys("username")
    driver.find_element_by_id("pwdInput").clear()
    driver.find_element_by_id("pwdInput").send_keys("password")
    driver.find_element_by_id("loginBtn").click()

# 退出
def logout():
    driver.find_element_by_link_text("退出").click()
    driver.quit()

driver = webdriver.Firefox()
driver.implicitly_wait(10)
driver.get("http://www.126.com")

login() # 登录

# 登录成功后的操作

# .....

logout() # 退出

# 退出后的操作

# 退出后的操作
```

## public.py

```
class Login():
    # 登录
    def user_login(self, driver):
        driver.find_element_by_id("idInput").clear()

        driver.find_element_by_id("idInput").send_keys("username")
        driver.find_element_by_id("pwdInput").clear()

        driver.find_element_by_id("pwdInput").send_keys("123456")
        driver.find_element_by_id("loginBtn").click()
    # 退出
    def user_logout(self, driver):
        driver.find_element_by_link_text("退出").click()
        driver.quit()

    """
    测试用例
    """
    def test_login(driver):
        driver.quit()
```

## mailTest.py

```
from selenium import webdriver
from public import Login
driver = webdriver.Firefox()
driver.implicitly_wait(10)
driver.get("http://www.126.com")
# 测试用例
```

```

Login().user_login(driver)
# .....
# .....
Login().user_logout(driver)

```

public.py Login() user\_login() user\_logout()

## 5.3

.....

### 5.3.1

.....

public.py

```

.....
# .....
def user_login(self, driver, username, password):

```

```

driver.find_element_by_id("idInput").clear()
driver.find_element_by_id("idInput").send_keys(username)
driver.find_element_by_id("pwdInput").clear()
driver.find_element_by_id("pwdInput").send_keys(password)
driver.find_element_by_id("loginBtn").click()

```

.....

```

def user_login():
    username = input("username:")
    password = input("password:")

```

mailTest.py

```

from selenium import webdriver
from public import Login
class LoginTest():
    def __init__(self):
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(10)
        self.driver.get("http://www.126.com")
    # admin
    def test_admin_login(self):
        username = 'admin'
        password = '123'
        Login().user_login(self.driver, username, password)
        self.driver.quit()
    # guest
    def test_guest_login(self):
        username = 'guest'

```

```

        password = '321'
        Login().user_login(self.driver, username, password)
        self.driver.quit()
LoginTest().test_admin_login()
LoginTest().test_guest_login()

```

在LoginTest类中，\_\_init\_\_()方法用于初始化URL，test\_admin\_login()和test\_guset\_login()方法分别用于测试管理员登录和访客登录。Login()类中的user\_login()方法用于用户登录。

## 5.3.2 百度搜索

本节将使用Selenium WebDriver实现百度搜索功能。首先，我们需要安装Selenium WebDriver。然后，我们将编写一个Python脚本，使用Selenium WebDriver驱动Firefox浏览器，访问百度搜索页面，输入搜索关键词，并点击搜索按钮。

baidu.py

```

from selenium import webdriver
search_text = ['python', '测试', 'text']
for text in search_text:
    driver = webdriver.Firefox()
    driver.implicitly_wait(10)
    driver.get("http://www.baidu.com")
    driver.find_element_by_id('kw').send_keys(text)
    driver.find_element_by_id('su').click()
    driver.quit()

```

search\_text for  
for  
for

### 5.3.3 txt

txt Python txt

- read() 读取全部
- readline() 读取一行
- readlines() 读取全部行

txt  
txt

user\_info.txt

zhangsan,123  
lisi,456  
wangwu,789

txt“,”

user\_info.py

```
user_file = open('user_info.txt', 'r')  
lines = user_file.readlines()  
user_file.close()
```

```

for line in lines:
    username = line.split(',')[0]
    password = line.split(',')[1]
print(username, password)

```

□□□□

```

=====          RESTART:          D:/pyse/user_info.py
=====

```

zhangsan 123

lisi 456

wangwu 789

```

    f=open("r")
    user_info.txt.readlines()
    txt.split()
    split()
    split()
    [0]
    [1]

```

```

    f.close()

```

## 5.3.4 csv

```

    f=open("r")
    user_info.txt
    CSV

```

```

    info.csv
    WPS
    Excel
    CSV
    5.4

```

00

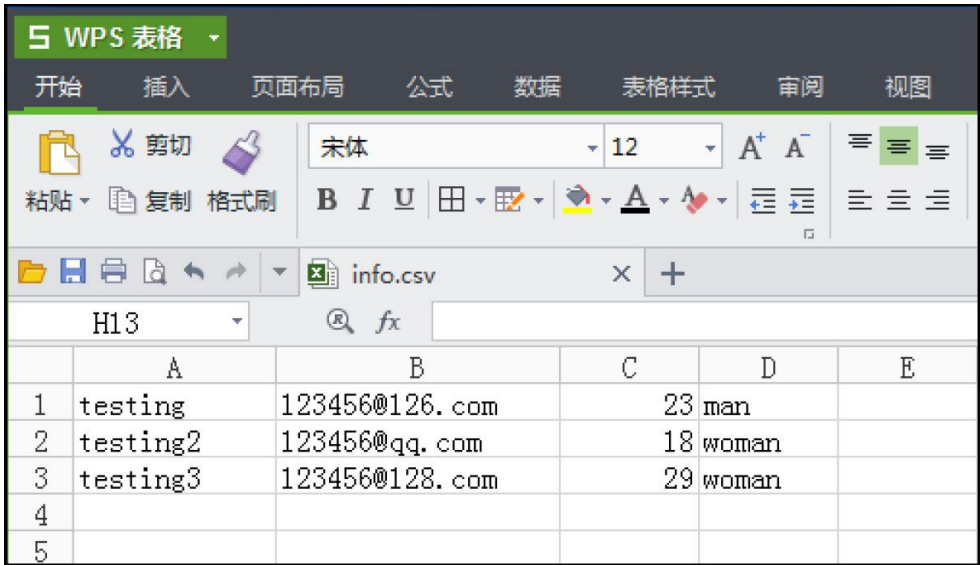


图5.4 CSV文件

编写csv\_read.py文件

csv\_read.py

```
import csv # 导入csv
# 读取 CSV 文件
date = csv.reader(open('info.csv', 'r'))
# 遍历数据
for user in date:
    print(user)

===== RESTART: D:/model/csv_read.py
=====
['testing', '123456@126.com', '23', 'man']
```



```
['testing2', '123456@qq.com', '18', 'woman']
['testing3', '123456@128.com', '29', 'woman']
```

```
    csv_reader = csv.reader(csv_file, delimiter=',')
    for row in csv_reader:
```

```
        print(row)
    print('Done')
```

csv\_read.py

```
import csv
data = csv.reader(open('info.csv', 'r'))
# 遍历数据
for user in data:
    print(user[1])
```

输出结果

```
===== RESTART: D:/model/csv_read.py
=====
123456@126.com
123456@qq.com
123456@128.com
```

```
    print('CSV文件读取成功')
    print('CSV文件名称: %s' % csv_file)
```

```
    print('CSV文件路径: %s' % csv_file)
    print('CSV文件内容: %s' % csv_content)
    print('CSV文件类型: %s' % csv_type)
    print('CSV文件大小: %s' % csv_size)
    print('CSV文件创建时间: %s' % csv_created_time)
    print('CSV文件最后修改时间: %s' % csv_modified_time)
    print('CSV文件所有者: %s' % csv_owner)
    print('CSV文件权限: %s' % csv_permissions)
    print('CSV文件内容: %s' % csv_content)
    print('CSV文件类型: %s' % csv_type)
    print('CSV文件大小: %s' % csv_size)
    print('CSV文件创建时间: %s' % csv_created_time)
    print('CSV文件最后修改时间: %s' % csv_modified_time)
    print('CSV文件所有者: %s' % csv_owner)
    print('CSV文件权限: %s' % csv_permissions)
```

## 5.3.5 使用xml

使用XML格式来描述数据，XML（Extensible Markup Language）是一种标记语言，用于描述结构化数据。XML文档通常通过URL来访问。

XML文档遵循1.6版本规范。

info.xml

```
<?xml version="1.0" encoding="utf-8"?>
<info>
  <base>
    <platform>Windows</platform>
    <browser>Firefox</browser>
    <url>http://www.baidu.com</url>
    <login username="admin" password="123456"/>
    <login username="guest" password="654321"/>
  </base>
  <test>
    <province>北京</province>
    <province>上海</province>
    <city>北京</city>
    <city>上海</city>
    <province>北京</province>
    <city>北京</city>
  </test>
</info>
```

info.xml XML

1

read\_xml.py

```
from xml.dom import minidom
# xml
dom = minidom.parse('info.xml')
# 
root = dom.documentElement
print(root.nodeName)
print(root.nodeValue)
print(root.nodeType)
print(root.ELEMENT_NODE)
```

```
===== RESTART: D:/model/read_xml.py
=====
```

info

None

1

1

xml minidom XML parse() XML  
documentElement XML

nodeType  
nodeName  
nodeValue  
nodeType

## 2

read\_xml.py

```
from xml.dom import minidom
# 读取xml
dom = minidom.parse('info.xml')
# 遍历xml
root = dom.documentElement
tagname = root.getElementsByTagName('browser')
print(tagname[0].tagName)
tagname = root.getElementsByTagName('login')
print(tagname[1].tagName)
tagname = root.getElementsByTagName('province')
print(tagname[2].tagName)

===== RESTART: D:/model/read_xml.py
=====

browser
login
province
```

`getElementByTagName()` 返回一个元素对象，通过该对象可以访问元素的属性。  
例如，通过 `login` 和 `province` 属性可以访问 `info.xml` 文件中的元素。  
如下所示：

- `getElementsByTagName('province')` 返回一个包含所有 `province` 元素的列表。
- `getElementsByTagName('province').tagName[0]` 返回第一个 `province` 元素的 `tagName` 属性。
- `getElementsByTagName('province').tagName[2]` 返回第三个 `province` 元素的 `tagName` 属性。

### 3 读取 XML 文件

`read_xml.py`

```
from xml.dom import minidom
# 读取 XML 文件
dom = minidom.parse('info.xml')
# 获取根元素
root = dom.documentElement
logins = root.getElementsByTagName('login')
# 获取第一个 login 元素的 username 属性
username = logins[0].getAttribute("username")
print(username)
# 获取第一个 login 元素的 password 属性
password = logins[0].getAttribute("password")
print(password)
```

```
# 从login中获取username
```

```
username = logins[1].getAttribute("username")
```

```
print(username)
```

```
# 从login中获取password
```

```
password = logins[1].getAttribute("password")
```

```
print(password)
```

```
=====
```

RESTART:

D:/model/read\_xml.py

```
=====
```

admin

123456

guest

654321

getAttribute() 从XML文档中获取元素的属性值 WebDriver 从XML文档中

get\_attribute() 从XML文档中

## 4 从XML文档中

read\_xml.py

```
from xml.dom import minidom
```

```
# 从XML文档中
```

```
dom = minidom.parse('info.xml')
```

```
# 从XML文档中
```

```
root = dom.documentElement
```

```
provinces = dom.getElementsByTagName('province')
```

```
citys = dom.getElementsByTagName('city')
```

```
# 遍历province
```

```
p2 = provinces[1].firstChild.data
```

```
print(p2)
```

```
# 遍历city
```

```
c1 = citys[0].firstChild.data
```

```
print(c1)
```

```
# 遍历city
```

```
c2 = citys[1].firstChild.data
```

```
print(c2)
```

```
=====
```

```
===== RESTART: D:/model/read_xml.py
```

```
=====
```

```
00
```

```
00
```

```
00
```

firstChild 遍历 data

WebDriver 文本

0000

遍历 province 遍历 city





## 第6章 Selenium IDE

---

本章主要介绍 Selenium IDE 的安装和配置。Selenium IDE 是一个基于 Firefox 的插件，用于录制和回放测试用例。本章将详细介绍如何安装 Selenium IDE，并配置其运行环境。

本章将介绍 Selenium IDE 的安装和配置。Selenium IDE 是一个基于 Firefox 的插件，用于录制和回放测试用例。本章将详细介绍如何安装 Selenium IDE，并配置其运行环境。

### 6.1 Selenium IDE

Selenium IDE 是一个基于 Firefox 的插件，用于录制和回放测试用例。本章将详细介绍如何安装 Selenium IDE，并配置其运行环境。Selenium IDE 的安装和配置非常简单，只需按照以下步骤操作即可。

安装 Selenium IDE

#### 6.1.1 安装

安装 Selenium IDE 的步骤如下：

<http://docs.seleniumhq.org/download/>

安装 Selenium IDE 的步骤如下：

## Selenium IDE

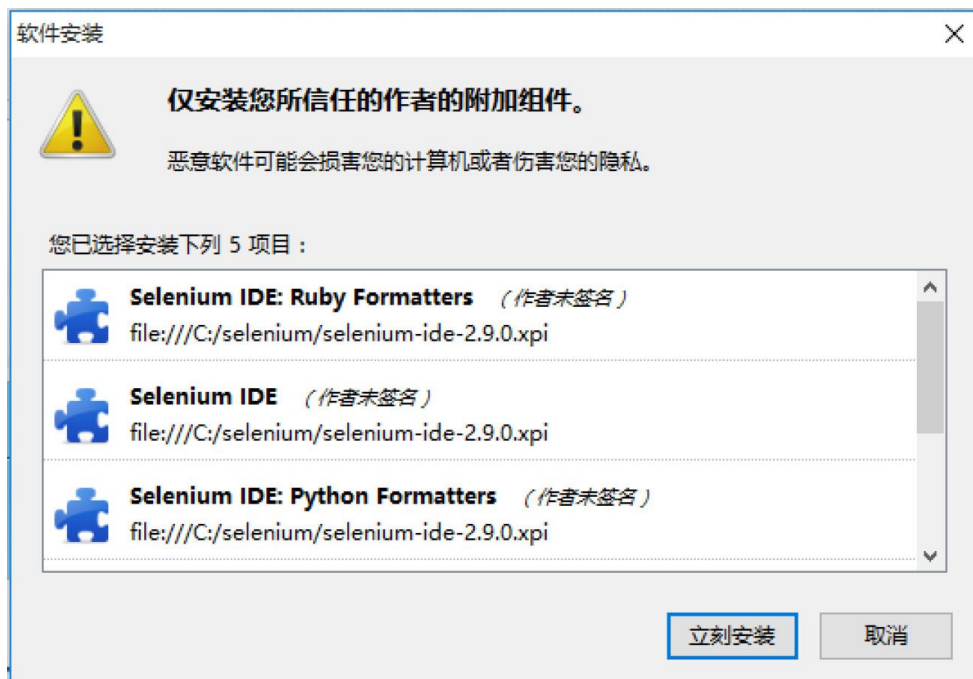
Selenium IDE is a Firefox plugin which records and plays back user interactions with the browser. Use this to either create simple scripts or assist in exploratory testing. It can also export Remote Control or WebDriver scripts, though they tend to be somewhat brittle and should be overhauled into some sort of Page Object-y structure for any kind of resiliency.

Download latest released version **2.9.0** released on 09/Mar/2015 or view the [Release Notes](#) and then [install some plugins](#).

Download previous version [2.8.0](#) released on 29/Sep/2014.

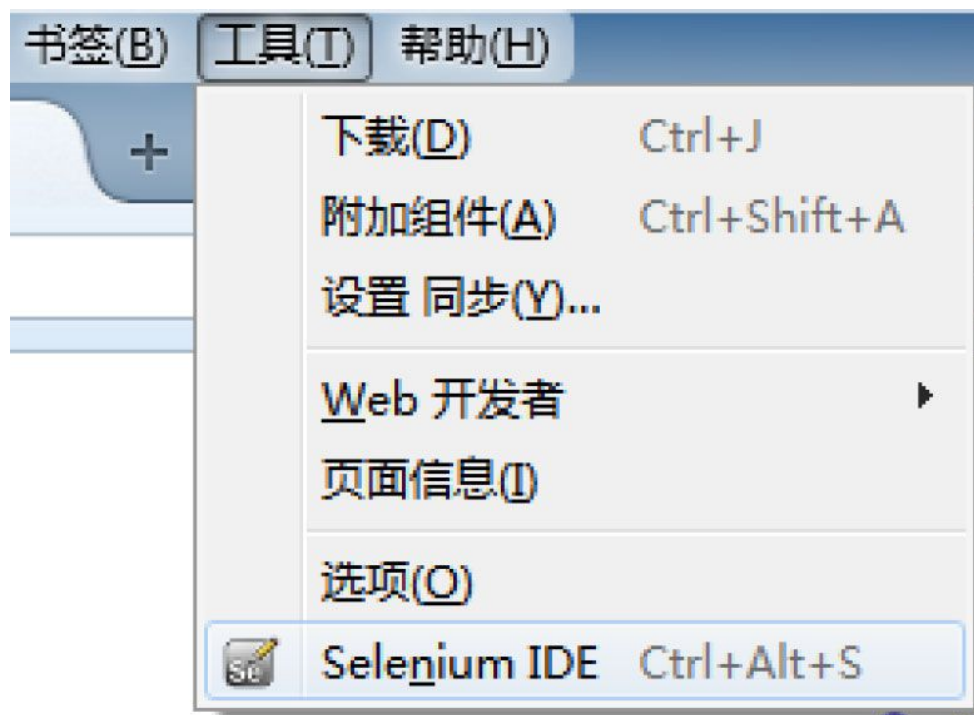
### 6.1 Selenium IDE

Firefox 6.2 Selenium IDE 6.2 “”  
“”



### 6.2 Firefox Selenium IDE

Firefox “” → “selenium IDE”  
Ctrl+Alt+S 6.3



### 6.3 Selenium IDE

### 6.1.2 ☐ ☐ ☐ ☐ ☐ ☐

6.1 Firefox Selenium IDE Selenium IDE selenium-ide-x.x.x.xpi

Firefoxのバージョンを「」→「」に変更する  
「」6.4



## 6.4 Firefox

selenium-ide-x.x.x.xpi “”

6.2 “”

# 6.2 Selenium IDE

Selenium IDE 6.5

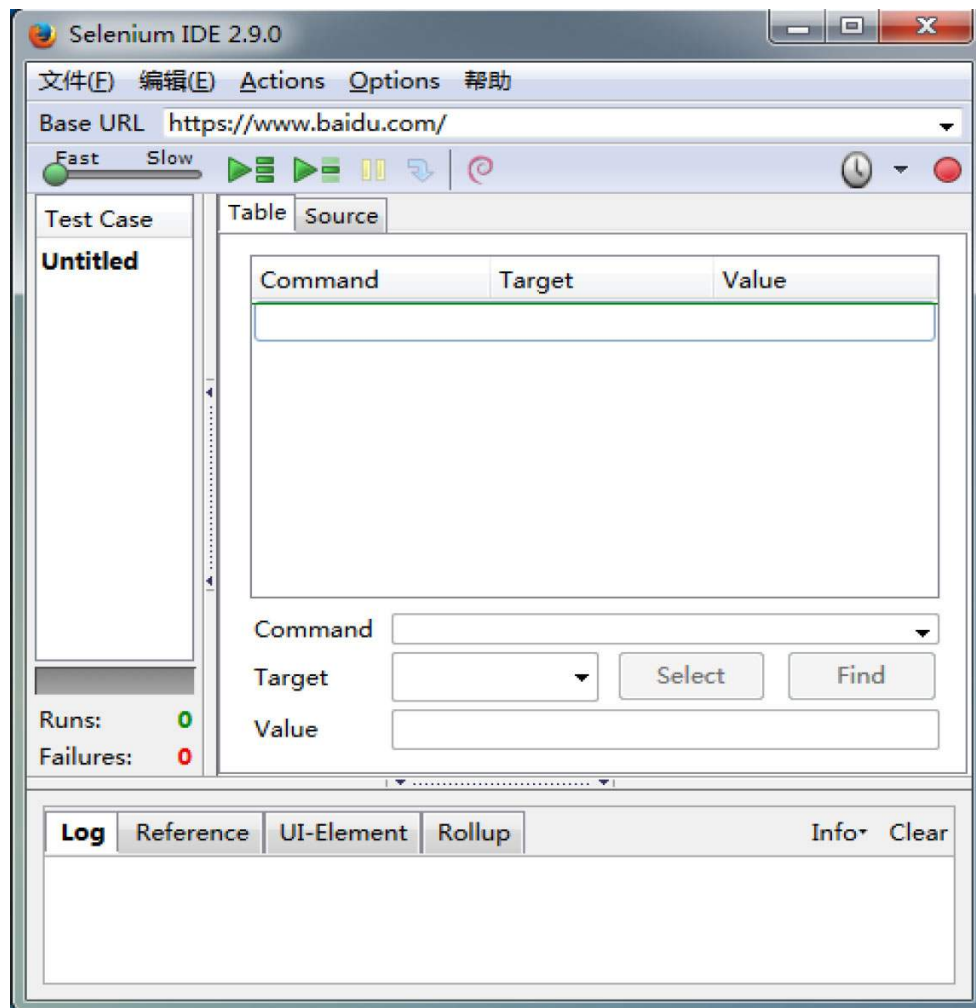


图6.5 Selenium IDE界面

Selenium IDE界面组成

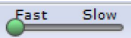






**1. 测试用例(F) 测试用例列表**

测试用例列表 测试用例列表

**Actions** 测试用例 测试用例列表

**Options** 测试用例 测试用例 Selenium IDE

**2. Base URL** 测试用例列表 URL

3     Fast   Slow 

4          

5          


6          










7          

8          

9          

10 **Test Case**      

11          

12 **Runs/Failures**          

13 **Table/Source**          

14 **Log/Reference/UI-Element/Rollup**       

**Log**          

**Reference**          

### 6.3.1 □□□□

Untitled (untitled suite) - Selenium IDE 2.9.0 \*

文件(F) 编辑(E) Actions Options 帮助

Base URL <https://www.baidu.com/>

Fast Slow

Test Case

**Untitled \***

| Command | Target | Value        |
|---------|--------|--------------|
| open    | /      |              |
| type    | id=kw  | selenium ide |
| click   | id=su  |              |

Command: open

Target: /

Value:

Runs: 0

Failures: 0

## 6.6 Selenium IDE

### 6.3.2 □□□□

Selenium IDEのインストールと起動方法について説明します。

## 1. Selenium IDEのインストール

TableのCommand、Target、Valueの列にテストケースの命令を入力します。図6.7を参照してください。

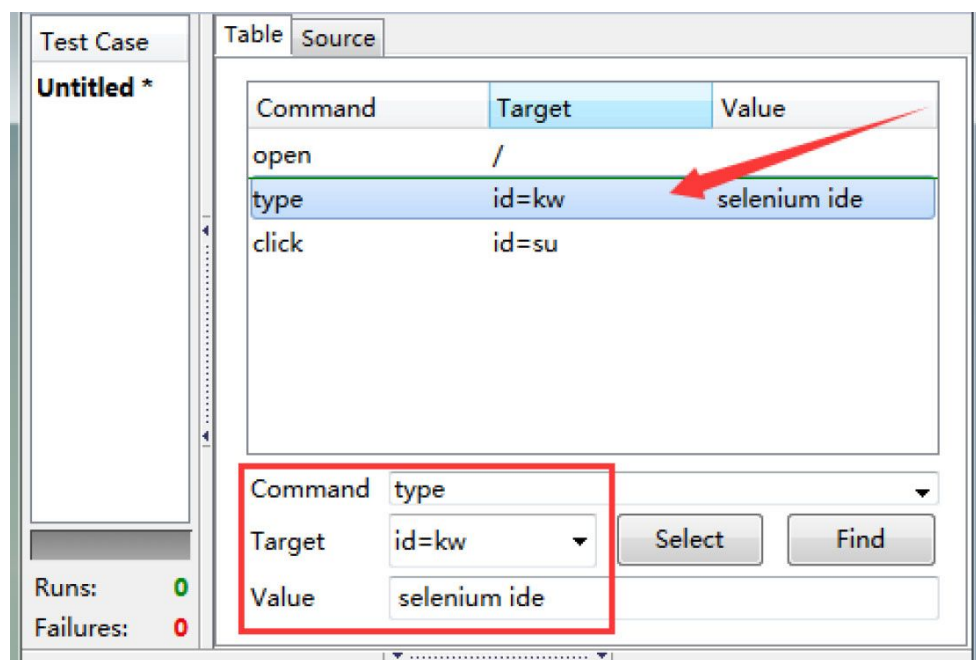


図6.7 Selenium IDEのスクリーンショット

## 2. テストケースの作成

右側のメニューから“Insert New Command”を選択して新しいテストケースの命令を追加します。図6.8を参照してください。



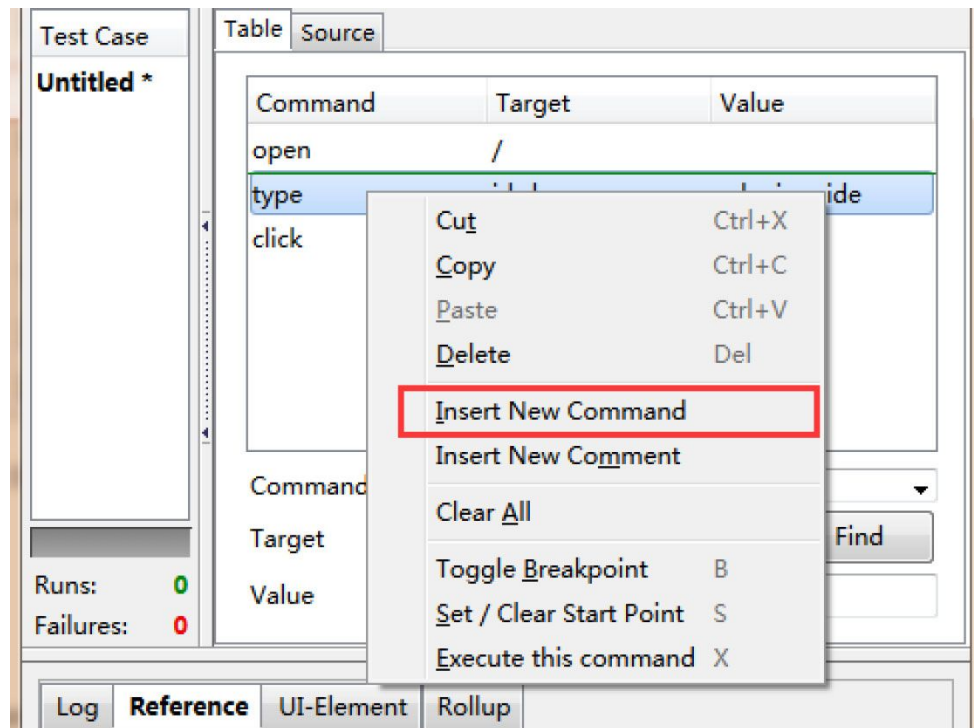


図6.8 右クリックメニュー

3. 実行

右クリックメニューから“Insert New Comment”を選択すると、コメントの入力欄が表示されます。

コメントの入力欄が表示された状態で、図6.9のように

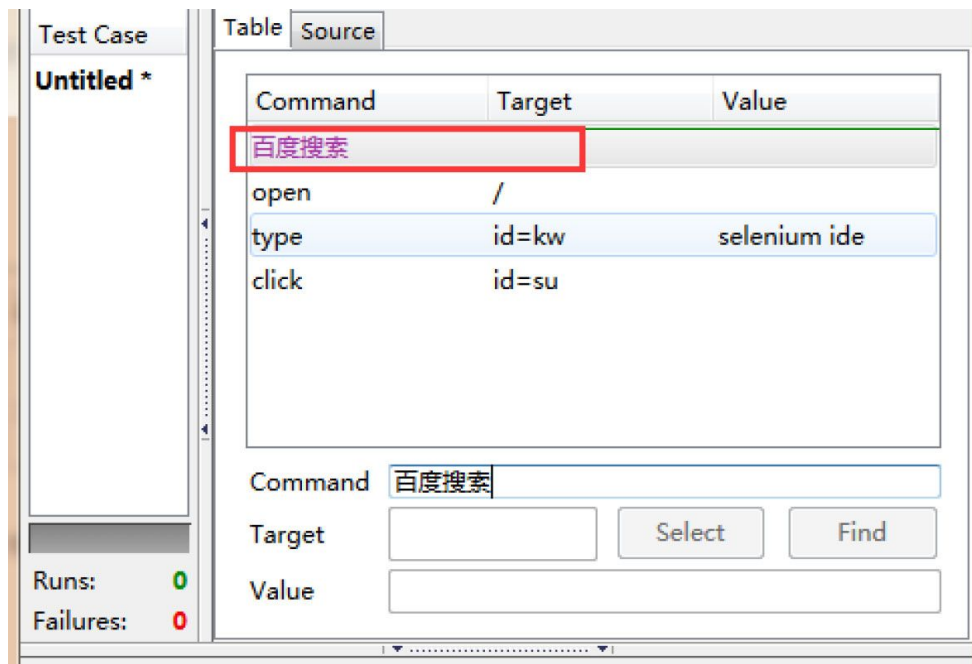
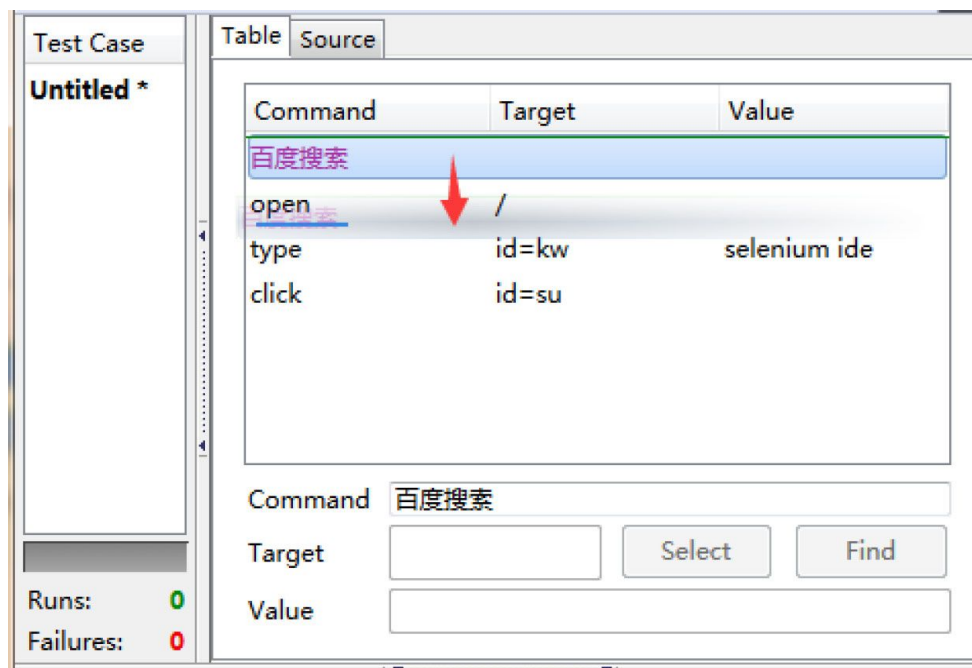


图6.9 百度搜索

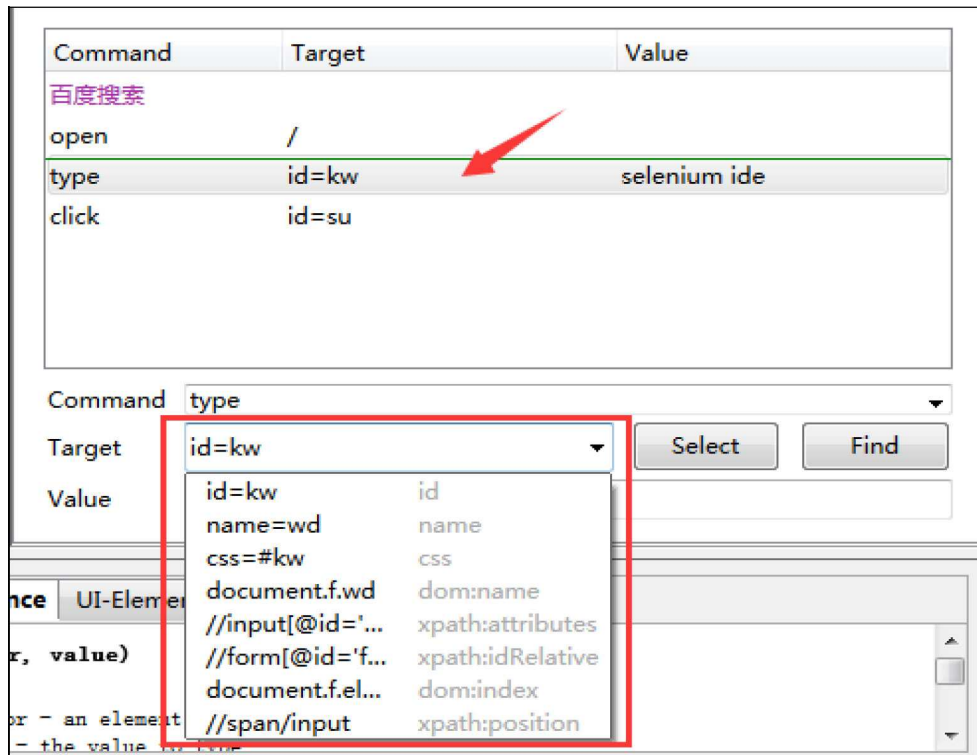
## 4. 百度搜索

百度搜索的测试用例如下所示，如图6.10所示。



## 5

Selenium IDE Target Selenium IDE Command Target 6.11



6.11

## 6.4 Selenium IDE

Selenium IDE Selenium IDE Command 6.12

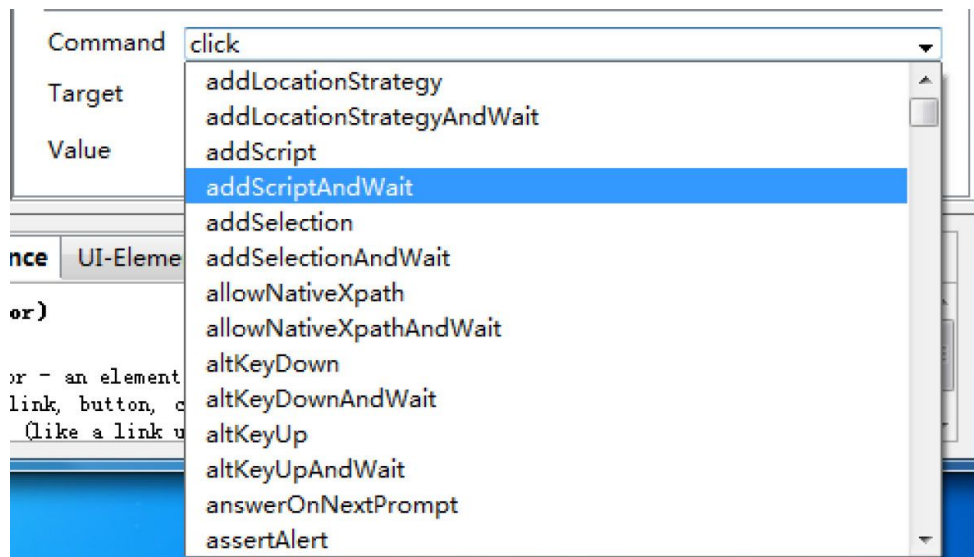


图6.12 Selenium IDE

## 1. 打开浏览器

```
open
open(url)
```

- 指定要打开的URL

```
open http://localhost/
```

| Command | Target            | Value |
|---------|-------------------|-------|
| open    | /mypage           |       |
| open    | http://localhost/ |       |

## 2-click

```
click(elementLocator)
```

- Selenium WebDriver
- Selenium WebDriver “clickAndWait”
- Selenium WebDriver JavaScript alert confirm verify assert Selenium

| Command      | Target       | Value |
|--------------|--------------|-------|
| click        | aCheckbox    |       |
| clickAndWait | submitButton |       |
| clickAndWait | anyLink      |       |

### 3□type

```
type(inputLocator.value)
```

- input
- 
- 

| Command | Target | Value |
|---------|--------|-------|
|         |        |       |

|             |                            |            |
|-------------|----------------------------|------------|
| type        | nameField                  | John Smith |
| typeAndWait | textBoxThatSubmitsOnChange | newValue   |

## 4 select

```
select(dropDownLocator,optionSpecifier)
```

- optionSpecifier is a string that identifies the option to select

- optionSpecifier can be a string "f\*b\*" where f is the first letter of the option text, b is the last letter of the option text, and \* is any number of letters in between

Examples:

| Command       | Target           | Value              |
|---------------|------------------|--------------------|
| select        | dropDown         | Australian Dollars |
| select        | dropDown         | index=0            |
| selectAndWait | currencySelector | value=AUD          |
| selectAndWait | currencySelector | label=Auslian D*rs |

## 5 goBack

```
goBack()
```

WebDriverCommand

| Command | Target | Value |
|---------|--------|-------|
| goBack  |        |       |

## 6 selectWindow

```
select(windowId)
```

- WebDriverCommand
- WebDriverTarget

| Command      | Target        | Value |
|--------------|---------------|-------|
| selectWindow | myPopupWindow |       |
| selectWindow | null          |       |

## 7 pause

```
pause(millisecnds)
```

- WebDriverCommandSelenium
- WebDriverTarget

| Command | Target | Value |
|---------|--------|-------|
|         |        |       |

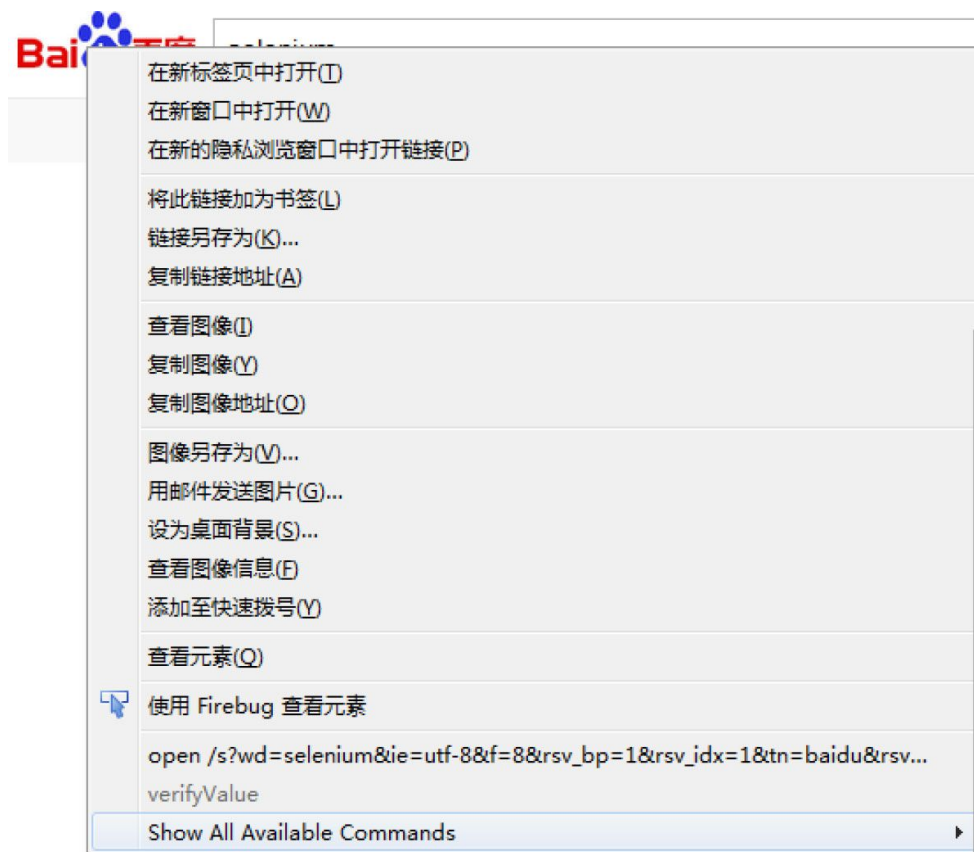




[illegible]

□ Selenium IDE □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

Selenium IDE Selenium IDE  
Firefox “Show All Available  
Commands” 6.13



□6.13 □□□□□□

□□□□□□□□□□□□□□□□6.14□□□

```

    6.14 4 assert verify waitFor
store 5

```

|   |
|---|
| open /s?wd=selenium&ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&rsv... |
| assertTitle selenium_百度搜索   |
| assertValue   |
| assertText css=img[alt="到百度首页"]                                     |
| assertTable   |
| assertElementPresent css=img[alt="到百度首页"]                           |
| verifyTitle selenium_百度搜索   |
| verifyValue   |
| verifyText css=img[alt="到百度首页"]                                     |
| verifyTable   |
| verifyElementPresent css=img[alt="到百度首页"]                           |
| waitForTitle selenium_百度搜索  |
| waitForValue  |
| waitForText css=img[alt="到百度首页"]                                    |
| waitForTable  |
| waitForElementPresent css=img[alt="到百度首页"]                          |
| storeTitle selenium_百度搜索  |
| storeValue  |
| storeText css=img[alt="到百度首页"]                                      |
| storeTable  |
| storeElementPresent css=img[alt="到百度首页"]                            |

□6.14 □□□□□□

- Title□ □□□□□□□□
- Value□ □□□□□□□□
- Text□ □□□□□□□□□□
- Table□ □□□□□□□□
- ElementPresent□ □□□□□□□□

## 6.5.1 □□

1. 打开浏览器
 2. 输入地址
 3. 输入关键词
 4. 点击搜索
 5. 断言标题
 6. 断言文本
 7. 断言元素存在
 8. 关闭浏览器

1. 打开浏览器
 2. 输入地址
 3. 输入关键词
 4. 点击搜索
 5. 断言标题
 6. 断言文本
 7. 断言元素存在
 8. 关闭浏览器

1. 打开浏览器
 2. 输入地址
 3. 输入关键词
 4. 点击搜索
 5. 断言标题
 6. 断言文本
 7. 断言元素存在
 8. 关闭浏览器

1. 打开浏览器
 2. 输入地址
 3. 输入关键词
 4. 点击搜索
 5. 断言标题
 6. 断言文本
 7. 断言元素存在
 8. 关闭浏览器

| Baidu Test Case      |                       |                 |
|----------------------|-----------------------|-----------------|
| Command              | Target                | Value           |
| open                 | http://www.baidu.com/ |                 |
| type                 | id=kw                 | selenium<br>ide |
| click                | id=su                 |                 |
| assertTitle          | selenium ide_百度       |                 |
| assertText           | css=img[alt="百度"]     |                 |
| assertElementPresent | css=img[alt="百度"]     |                 |
| close                |                       |                 |

1. 打开浏览器
 2. 输入地址
 3. 输入关键词
 4. 点击搜索
 5. 断言标题
 6. 断言文本
 7. 断言元素存在
 8. 关闭浏览器

6.5.2 百度

1. 打开百度首页
 2. 输入关键词
 3. 点击搜索按钮
 4. 验证搜索结果

6.14 百度测试用例

| Baidu Test Case      |                       |                 |
|----------------------|-----------------------|-----------------|
| Command              | Target                | Value           |
| open                 | http://www.baidu.com/ |                 |
| type                 | id=kw                 | selenium<br>ide |
| click                | id=su                 |                 |
| verifyTitle          | selenium ide_百度       |                 |
| verifyText           | css=img[alt="百度"]     |                 |
| verifyElementPresent | css=img[alt="百度"]     |                 |
| close                |                       |                 |

1. 打开百度首页
 2. 输入关键词
 3. 点击搜索按钮
 4. 验证搜索结果

百度测试用例

| Baidu Test Case |                       |       |
|-----------------|-----------------------|-------|
| Command         | Target                | Value |
| open            | http://www.baidu.com/ |       |

|               |                          |                    |
|---------------|--------------------------|--------------------|
| type          | id=kw                    | selenium id        |
| click         | id=su                    |                    |
| a ssert Title | selenium id_[] [] [] sss |                    |
| type          | id=kw                    | selenium webdriver |
| click         | id=su                    |                    |
| close         |                          |                    |

在代码中，我们使用 `selenium id_[] [] [] sss` 来指定断言的 ID。
 在 Selenium IDE 中，我们使用 `log` 来记录断言的结果。

```

• [info] Playing test case baidu_case
• [info] Executing: |open | http://www.baidu.com | |
• [info] Executing: |type | id=kw | selenium id |
• [info] Executing: |click | id=su | |
• [info] Executing: |pause | 2000 | |
• [info] Executing: |assertTitle | selenium id_[] [] [] sss | |
  • [error] Actual value 'selenium id_[] [] [] ' did not match 'selenium id_[] [] [] sss'
• [info] Test case failed
• [info] Test suite completed: 1 played, 1 failed
  
```

在代码中，我们使用 `assertTitle` 来指定断言的 ID。
 在 Selenium IDE 中，我们使用 `verifyTitle` 来记录断言的结果。

```

• [info] Playing test case baidu_case
• [info] Executing: |open | http://www.baidu.com | |
  
```

```

    • [info] Executing: |type | id=kw | selenium id |
    • [info] Executing: |click | id=su | |
    • [info] Executing: |pause | 2000 | |
    • [info] Executing: |verifyTitle | selenium id_[] [] [] sss | |
      • [error] Actual value 'selenium id_[] [] [] ' did not match
'selenium id_[] [] [] sss'

    • [info] Executing: |type | id=kw | selenium webdriver |
    • [info] Executing: |click | id=su | |
    • [info] Executing: |close | | |
    • [info] Test case failed
    • [info] Test suite completed: 1 played, 1 failed

```

“verifyTitle”

## 6.6 ☐☐☐☐☐

6.14 waitForStore

### 6.6.1 ☐ ☐

Selenium IDE pause waitFor

| Baidu Test Case |                       |       |
|-----------------|-----------------------|-------|
| Command         | Target                | Value |
| open            | http://www.baidu.com/ |       |

|                       |                      |                 |
|-----------------------|----------------------|-----------------|
|                       |                      |                 |
| type                  | id=kw                | selenium<br>ide |
| click                 | id=su                |                 |
| waitForTitle          | selenium ide_□□□□    |                 |
| waitForText           | css=img[alt="□□□□□"] |                 |
| waitForElementPresent | css=img[alt="□□□□□"] |                 |
| close                 |                      |                 |

waitFor□Value□□□□□□□□□□60□□□□□□□□waitForTitle□  
 waitForText□waitForElementPresent□□□□□□□□□□□□□□□□□□  
 □□

## 6.6.2 □□

store□□□□□□□

| Baidu Test Case |                       |                 |
|-----------------|-----------------------|-----------------|
| Command         | Target                | Value           |
| open            | http://www.baidu.com/ |                 |
| type            | id=kw                 | selenium<br>ide |
| click           | id=su                 | □□□□            |
|                 |                       |                 |

|                        |                     |         |
|------------------------|---------------------|---------|
| storeTitle             | selenium id_百度搜索    | title   |
| storeText              | css=img[alt="百度搜索"] | text    |
| storeForElementPresent | css=img[alt="百度搜索"] | element |
| close                  |                     |         |

百度搜索  
title百度搜索  
text百度搜索  
element百度搜索

| Baidu Test Case        |                       |                |
|------------------------|-----------------------|----------------|
| Command                | Target                | Value          |
| open                   | http://www.baidu.com/ |                |
| type                   | id=kw                 | selenium<br>id |
| waitForValue           | id=su                 |                |
| click                  | id=su                 | 搜索             |
| pause                  | 2000                  |                |
| storeTitle             | selenium id_百度搜索      | title          |
| storeText              | css=img[alt="百度搜索"]   | text           |
| storeForElementPresent | css=img[alt="百度搜索"]   | element        |
| verifyTitle            | selenium id_百度搜索      | title          |
| verifyText             | css=img[alt="百度搜索"]   | text           |
|                        |                       |                |



|                      |                      |         |
|----------------------|----------------------|---------|
| assertElementPresent | css=img[alt="image"] | element |
| close                |                      |         |

store() is used to store the value of an expression in a variable.

store()

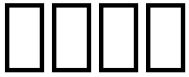
```
store(expression, variableName)
```

- expression is the value to be stored in the variableName.

| Command | Target                                     | Value    |
|---------|--|----------|
| store   | Mr John Smith                              | fullname |
| store   | \$.title\$.firstname\$.<br>suname          | fullname |
| store   | javascript.<br>Math.round(Math.PI*100)/100 | PI       |

store() is used to store the value of an expression in a variable.

| Command | Target       | Value     |
|---------|--------------|-----------|
| store   | selenium ide | value     |
| type    | id=kw        | \${value} |



```

Selenium IDE
Selenium IDE

```

[illegible]



## 第7章 unittest

---

unittest 是 Python 标准库中用于编写单元测试的框架。它提供了丰富的 API 来支持各种测试需求，包括断言、测试套件、测试用例、测试 runner 等。

unittest 框架支持 Web 应用的测试。通过编写测试用例，可以验证 Web 应用的行为是否符合预期。unittest 提供了丰富的断言方法，如 `assertEqual`、`assertNotEqual`、`assertTrue`、`assertFalse` 等，用于验证测试结果的预期值。

unittest 框架还支持测试套件（`unittest.TestCase`）的继承。通过继承 `unittest.TestCase`，可以方便地编写测试用例。测试套件可以包含多个测试用例，并且可以指定测试套件的组织结构。

unittest 框架还支持测试 runner 的自定义。通过编写自定义的测试 runner，可以实现对测试结果的自定义输出。unittest 提供了丰富的 API 来支持测试 runner 的自定义，如 `unittest.runner.TextRunner`、`unittest.runner.VerboseRunner` 等。

unittest 框架还支持测试结果的报告。通过编写自定义的报告生成器，可以实现对测试结果的自定义报告。unittest 提供了丰富的 API 来支持报告生成器的自定义，如 `unittest.runner.TextRunner`、`unittest.runner.VerboseRunner` 等。

unittest 框架还支持 Web 应用的测试。通过编写测试用例，可以验证 Web 应用的行为是否符合预期。unittest 提供了丰富的断言方法，如 `assertEqual`、`assertNotEqual`、`assertTrue`、`assertFalse` 等，用于验证测试结果的预期值。

### 7.1 unittest

unittest 是 Python 标准库中用于编写单元测试的框架。它提供了丰富的 API 来支持各种测试需求，包括断言、测试套件、测试用例、测试 runner 等。Python 社区中还有许多其他的测试框架，如 `doctest`、`unittest`、`pytest`、`nose` 等。其中，`unittest` 是最常用的测试框架之一，因为它提供了丰富的 API 来支持各种测试需求，并且与 Python 标准库紧密集成。

Python unittest Python 2.1 unittest Python Python

## 7.1.1

calculator.py

calculator.py

#

class Count:

def \_\_init\_\_(self, a, b):

self.a = int(a)

self.b = int(b)

#

def add(self):

return self.a + self.b

Count \_\_init\_\_() add()

test.py

test.py

```

from calculator import Count
# 测试用例
class TestCount:
    def test_add(self):
        try:
            j = Count(2, 3)
            add = j.add()
            assert(add == 5), 'Integer addition result error!'
        except AssertionError as msg:
            print(msg)
        else:
            print('Test pass!')
# 测试用例
mytest=TestCount()
mytest.test_add()

```

从calculator模块导入Count模块的test\_add()方法，创建Count对象，调用add()方法，断言结果等于5，否则抛出AssertionError异常，并打印异常信息。

## Python Shell

```

# assert测试用例
===== RESTART: D:/test/test.py
=====

Test pass!
# assert测试用例

```

=====

RESTART:

D:/test/test.py

Integer addition result error!

test.py

unittest

test.py

```
from calculator import Count
import unittest
class TestCount(unittest.TestCase):
    def setUp(self):
        print("test start")
    def test_add(self):
        j = Count(2, 3)
        self.assertEqual(j.add(), 5)
    def tearDown(self):
        print("test end")
if __name__ == '__main__':
    unittest.main()
```

unittest.TestCase unittest.TestCase

setUp() "test start"  
tearDown() setUp() "test end"

test\_add() Count add() unittest.assertEqual() add() assertEquals() TestCase

unittest.main() TestLoader "test"

## Python

```
if __name__ == '__main__':  
    Python  
    1 Python.py  
    2.py  
    3 Python import  
  
    if __name__ == '__main__':  
        Python.py
```



```
__main__ if __name__ == '__main__':
    unittest.main()
```

## 7.1.2 测试框架

unittest 框架包含 4 个主要组件：test fixture、test case、test suite、test runner。

### 1 Test Case

TestCase 类定义了 setUp()、run()、tearDown() 等方法。其中 setUp() 方法在测试用例执行前调用，tearDown() 方法在测试用例执行后调用。run() 方法用于执行测试用例。unittest 框架通过 TestCase 类来组织测试用例。

### 2 Test Suite

TestSuite 类用于组织多个 Test Case。可以通过 addTest() 方法将 Test Case 添加到 TestSuite 中。TestSuite 类还提供了 run() 方法来执行所有包含的 Test Case。

### 3 Test Runner

unittest.TestCase 클래스의 run() 메서드는 test suite/test case를 test runner로 실행합니다. unittest.TestCase 클래스의 setUp()와 tearDown() 메서드는 각각 테스트 시작 전과 테스트 후 실행되는 메서드입니다.

## 4 Test Fixture

unittest.TestCase 클래스의 setUp()와 tearDown() 메서드는 각각 테스트 시작 전과 테스트 후 실행되는 메서드입니다. setUp() 메서드는 테스트 시작 전 실행되는 메서드이고, tearDown() 메서드는 테스트 후 실행되는 메서드입니다.

tearDown 메서드는 test case 실행 후 실행되는 메서드입니다.

test.py

```
test.py

from calculator import Count
import unittest

class TestCount(unittest.TestCase):
    def setUp(self):
        print("test start")
    def test_add(self):
        j = Count(2, 3)
```

```

        self.assertEqual(j.add(), 5)
def test_add2(self):
    j = Count(41, 76)
    self.assertEqual(j.add(), 117)
def tearDown(self):
    print("test end")
if __name__ == '__main__':
    # 测试
    suite = unittest.TestSuite()
    suite.addTest(TestCount("test_add2"))
    # 运行
    runner = unittest.TextTestRunner()
    runner.run(suite)

#####test_add2()#####
#####main()#####
#####

```

使用unittest模块的TestSuite()方法添加测试用例，使用TextTestRunner()方法运行测试套件。

## Python Shell

```

=====
RESTART:          D:/test/test.py
=====
test start

```

```
test end
.
-----
-----
Ran 1 test in 0.016s
OK
```

unittest.TestCase.setUp/tearDown

### 7.1.3 unittest.TestCase

unittest.TestCase 是 Python 中用于编写测试的类。它提供了许多断言方法，用于验证测试的结果是否符合预期。unittest.TestCase 是 unittest 模块的一部分，该模块是 Python 标准库中的一个模块。

| 方法                   | 描述               | 版本  |
|----------------------|------------------|-----|
| assertEqual(a, b)    | a == b           |     |
| assertNotEqual(a, b) | a != b           |     |
| assertTrue(x)        | bool(x) is True  |     |
| assertFalse(x)       | bool(x) is False |     |
| assertIs(a, b)       | a is b           | 3.1 |
| assertIsNot(a, b)    | a is not b       | 3.1 |
| assertIsNone(x)      | x is None        | 3.1 |
| assertIsNotNone(x)   | x is not None    | 3.1 |

|  |                                   |     |
|--|-----------------------------------|-----|
| <code>assertIn(a, b)</code>            | <code>a in b</code>               | 3.1 |
| <code>assertNotIn(a, b)</code>         | <code>a not in b</code>           | 3.1 |
| <code>assertIsInstance(a, b)</code>    | <code>isinstance(a, b)</code>     | 3.2 |
| <code>assertNotIsInstance(a, b)</code> | <code>not isinstance(a, b)</code> | 3.2 |

- `assertEqual(first, second, msg=None)`

`msg` is a string that will be used in the error message if the assertion fails.  
 If `msg` is `None`, the default error message will be used.

`test.py`

```

import unittest

class Test(unittest.TestCase):
    def setUp(self):
        number = input("Enter a number:")
        self.number = int(number)

    def test_case(self):
        self.assertEqual(self.number, 10, msg="Your input is not 10!")

    def tearDown(self):
        pass

if __name__ == "__main__":
    unittest.main()

```

Python IDLE

```
setUp()
test_case()
assertEqual()
10
msg
```

## Python Shell

```
===== RESTART: D:/test/test.py
=====
Enter a number:12
F
=====
=====
FAIL: test_case (__main__.Test)
-----
-----
Traceback (most recent call last):
  File "D:\test\test.py", line 11, in test_case
    self.assertEqual(self.number, 10, msg="Your input is not
10!")
AssertionError: 12 != 10 : Your input is not 10!
-----
-----
Ran 1 test in 3.760s
FAILED (failures=1)
```

assertNotEqual(12, 10, msg="Your input is not 10!")

```
- assertNotEqual(first, second, msg=None)
```

assertNotEqual() assertEqual() assert() assertNot()

```
- assertTrue(expr, msg=None)
```

```
- assertFalse(expr, msg=None)
```

assertTrue() assertFalse()

count.py

count.py

```
# is_prime.py
```

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
```

is\_prime() returns True if n is a prime number, False otherwise. For example, is\_prime(2) returns True, is\_prime(1) returns False, is\_prime(4) returns False.

## test.py

```
from count import is_prime
import unittest
class Test(unittest.TestCase):
    def setUp(self):
        print("test start")
    def test_case(self):
        self.assertTrue(is_prime(7), msg="Is not prime!")
    def tearDown(self):
        print("test end")
if __name__ == "__main__":
    unittest.main()

    is_prime()
    assertTrue()True
```

- assertIn(first, second, msg=None)
- assertNotIn(first, second, msg=None)

## test.py

```
import unittest
class Test(unittest.TestCase):
    def setUp(self):
        print("test start")
```



```

def test_case(self):
    a = "hello"
    b = "hello world"
    self.assertIn(a, b, msg="a is not in b")
def tearDown(self):
    print("test end")
if __name__ == "__main__":
    unittest.main()

```

```

assertIn(a, b, msg="a is not in b")

```

- assertIs(first, second, msg=None)
- assertIsNot(first, second, msg=None)

```

assertEqual(a, b, msg="a is not equal to b")

```

- assertIsNone(expr, msg=None)
- assertIsNotNone(expr, msg=None)

```

assertNone(expr, msg=None)

```

- assertIsInstance(obj, cls, msg=None)
- assertNotIsInstance(obj, cls, msg=None)

```

assertIsInstance(obj, cls, msg=None)

```

unittest  
Python unittest

## 7.1.4

unittest

7.1.1 calculator.py sub()

calculator.py

```
#
class Count():
    def __init__(self, a, b):
        self.a = int(a)
        self.b = int(b)
    #
    def add(self):
        return self.a + self.b
    #
    def sub(self):
        return self.a - self.b
```

calculator sub  
test.py

## test.py

```
from calculator import Count
import unittest

class TestAdd(unittest.TestCase):
    def setUp(self):
        print("test add start")
    def test_add(self):
        j = Count(2, 3)
        self.assertEqual(j.add(), 5)
    def test_add2(self):
        j = Count(41, 76)
        self.assertEqual(j.add(), 117)
    def tearDown(self):
        print("test add end")

class TestSub(unittest.TestCase):
    def setUp(self):
        print("test sub start")
    def test_sub(self):
        j = Count(2, 3)
        self.assertEqual(j.sub(), -1)
    def test_sub2(self):
        j = Count(71, 46)
        self.assertEqual(j.sub(), 25)
    def tearDown(self):
        print("test sub end")

if __name__ == '__main__':
```

```
# 测试套件
suite = unittest.TestSuite()
suite.addTest(TestAdd("test_add"))
suite.addTest(TestAdd("test_add2"))
suite.addTest(TestSub("test_sub"))
suite.addTest(TestSub("test_sub2"))

# 测试运行器
runner = unittest.TextTestRunner()
runner.run(suite)
```

TestAdd()TestSub()calculator.py  
 add()sub()TestSuiteaddTest()  
 测试套件测试运行器

## Python Shell

```
===== RESTART: D:/test/test.py
=====
test add start
test add end
.test add start
test add end
.test sub start
test sub end
.test sub start
test sub end
.
-----
```

-----

Ran 4 tests in 0.047s

OK

```
setUp()tearDown()
setUp()tearDown()

```

test.py

```
from calculator import Count
import unittest
class MyTest(unittest.TestCase):
    def setUp(self):
        print("test case start")
    def tearDown(self):
        print("test case end")
class TestAdd(MyTest):
    def test_add(self):
        j = Count(2, 3)
        self.assertEqual(j.add(), 5)
    def test_add2(self):
        j = Count(41, 76)
        self.assertEqual(j.add(), 117)
class TestSub(MyTest):
    def test_sub(self):
        j = Count(2, 3)
        self.assertEqual(j.sub(), -1)
```

```

def test_sub2(self):
    j = Count(71, 46)
    self.assertEqual(j.sub(), 25)
if __name__ == '__main__':
    unittest.main()

```

class MyTest(unittest.TestCase):
 def setUp(self):
 def tearDown(self):
 def test\_sub2(self):
 def setUp(self):
 def tearDown(self):

## 7.1.5 discover

unittest.TestCase
 unittest.TestCase
 test.py
 unittest.TestCase

test.py

testpro/

└─ count.py

└─ testadd.py

└─ testsub.py

└─ runtest.py

□□□□□□□□□□

testadd.py

```
from calculator import Count
import unittest
class TestAdd(unittest.TestCase):
    def setUp(self):
        print("test case start")
    def tearDown(self):
        print("test case end")
    def test_add(self):
```

```

        j = Count(2, 3)
        self.assertEqual(j.add(), 5)
    def test_add2(self):
        j = Count(41, 76)
        self.assertEqual(j.add(), 117)
if __name__ == '__main__':
    unittest.main()

```

### testsub.py

```

from calculator import Count
import unittest
class TestSub(unittest.TestCase):
    def setUp(self):
        print("test case start")
    def tearDown(self):
        print("test case end")
    def test_sub(self):
        j = Count(2, 3)
        self.assertEqual(j.sub(), -1)
    def test_sub2(self):
        j = Count(71, 46)
        self.assertEqual(j.sub(), 25)
if __name__ == '__main__':
    unittest.main()

```

□□□□□□□□□□runtest.py□□□



## runtest.py

```
import unittest
# unittest
import testadd
import testsub
# unittest
suite = unittest.TestSuite()
suite.addTest(testadd.TestAdd("test_add"))
suite.addTest(testadd.TestAdd("test_add2"))
suite.addTest(testsub.TestSub("test_sub"))
suite.addTest(testsub.TestSub("test_sub2"))
if __name__ == '__main__':
    # unittest
    runner = unittest.TextTestRunner()
    runner.run(suite)

    unittest.TextTestRunner().run(suite)

    unittest.TextTestRunner().run(suite)
runtest.py unittest.addTest() unittest/ unittest unittest
unittest unittest unittest unittest unittest TestLoader unittest
discover() unittest unittest
```

TestLoader

unittest.defaultTestLoader.discover()

```
discover(start_dir=pattern='test*.py' top_level_dir=None)
```

- start\_dir
- pattern='test\*.py' “test” “.py” “\*”
- top\_level\_dir=None None

discover() runtest.py

runtest.py

```
import unittest
#
test_dir = './'
discover = unittest.defaultTestLoader.discover(test_dir,
pattern='test*.py')
if __name__ == '__main__':
    runner = unittest.TextTestRunner()
    runner.run(discover)
```

discover() test\_dir test\*.py run() discover

unittest

## 7.2 unittest

unittest 7.1  
unittest

### 7.2.1

unittest  
unittest  
unittest

unittest

test.py

```
import unittest
class TestBdd(unittest.TestCase):
    def setUp(self):
        print("test TestBdd :")
    def test_ccc(self):
        print("test ccc")
    def test_aaa(self):
        print("test aaa")
    def tearDown(self):
```

```

        pass
class TestAdd(unittest.TestCase):
    def setUp(self):
        print("test TestAdd :")
    def test_bbb(self):
        print("test bbb")
    def tearDown(self):
        pass
if __name__ == '__main__':
    unittest.main()

```

□□□□□□□□□□

## Python Shell

```

=====                                RESTART:                D:/test/test.py
=====

test TestAdd :
test bbb
.test TestBdd :
test aaa
.test TestBdd :
test ccc
.
-----
-----

Ran 3 tests in 0.047s
OK

```

unittest  
unittest

unittest ASCII 0~9 A~Z  
a~z TestAdd TestBdd test\_aaa()  
test\_ccc()

unittest

test\_ccc() main()  
TestSuite addTest()

test.py

```
.....  
if __name__ == '__main__':  
    #  
    suite = unittest.TestSuite()  
    suite.addTest(TestBdd("test_ccc"))  
    suite.addTest(TestAdd("test_bbb"))  
    suite.addTest(TestBdd("test_aaa"))  
    #  
    runner = unittest.TextTestRunner()  
    runner.run(suite)
```

Python Shell

```
=====
=====
```

RESTART:

D:/test/test.py

```
test TestBdd :
test ccc
.test TestAdd :
test bbb
.test TestBdd :
test aaa
.
```

```
-----
-----
```

Ran 3 tests in 0.037s

OK

```
    addTest()discover()
main()
    "test_a"    "test_z"
```

## 7.2.2

```
Web

```

test\_project/test\_case/

```
├── test_bbb/
│   ├── test_ccc/
│   │   └── test_c.py
│   └── test_b.py
├── test_ddd/
│   └── test_d.py
└── test_a.py
```

unittest 模块的 discover() 方法从 start\_dir 开始递归地搜索子目录，并返回所有匹配的测试用例。在 test\_a.py 中，我们使用 unittest 模块的 discover() 方法来发现 test\_case/ 目录下的所有测试用例。\_\_init\_\_.py 文件是 Python 3.6 版本引入的。

## 7.2.3 测试用例的跳过

在某些情况下，我们可能希望跳过某些测试用例。unittest 模块提供了 skip() 和 skipIf() 方法来跳过测试用例。

- unittest.skip(reason)

unittest.skip(reason) 方法用于跳过测试用例。

- unittest.skipIf(condition, reason)

unittest.TestCase

- `unittest.skipUnless(condition, reason)`

unittest.TestCase

- `unittest.expectedFailure()`

unittest.TestCase

test.py

```
import unittest

class MyTest(unittest.TestCase):
    def setUp(self):
        pass
    def tearDown(self):
        pass
    @unittest.skip("unittest.skip")
    def test_skip(self):
        print("test aaa")
    @unittest.skipIf(3 > 2, "unittest.skipIf True")
    def test_skip_if(self):
        print('test bbb')
    @unittest.skipUnless(3 > 2, "unittest.skipUnless True")
    def test_skip_unless(self):
        print('test ccc')
    @unittest.expectedFailure
    def test_expected_failure(self):
```



|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|

=====



.....

.....

0K (skipped=2, expected failures=1)

[illegible]

```
@unittest.skip(" ")
```

■■■■■

```
import unittest

def setUpModule():
    print("test module start >>>>>>>>>>>>>>>")

def tearDownModule():
    print("test module end >>>>>>>>>>>>>>>")

class Test(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        print("test class start =====>")
    @classmethod
    def tearDownClass(cls):
        print("test class end =====>")
    def setUp(self):
        print("test case start -->")
    def tearDown(self):
        print("test case end -->")
```

□ □ □ □ □ □ □

[illegible]

setUpModule/tearDownModule 模块级别的setUp/tearDown

setUpClass/tearDownClass 类级别的setUp/tearDown

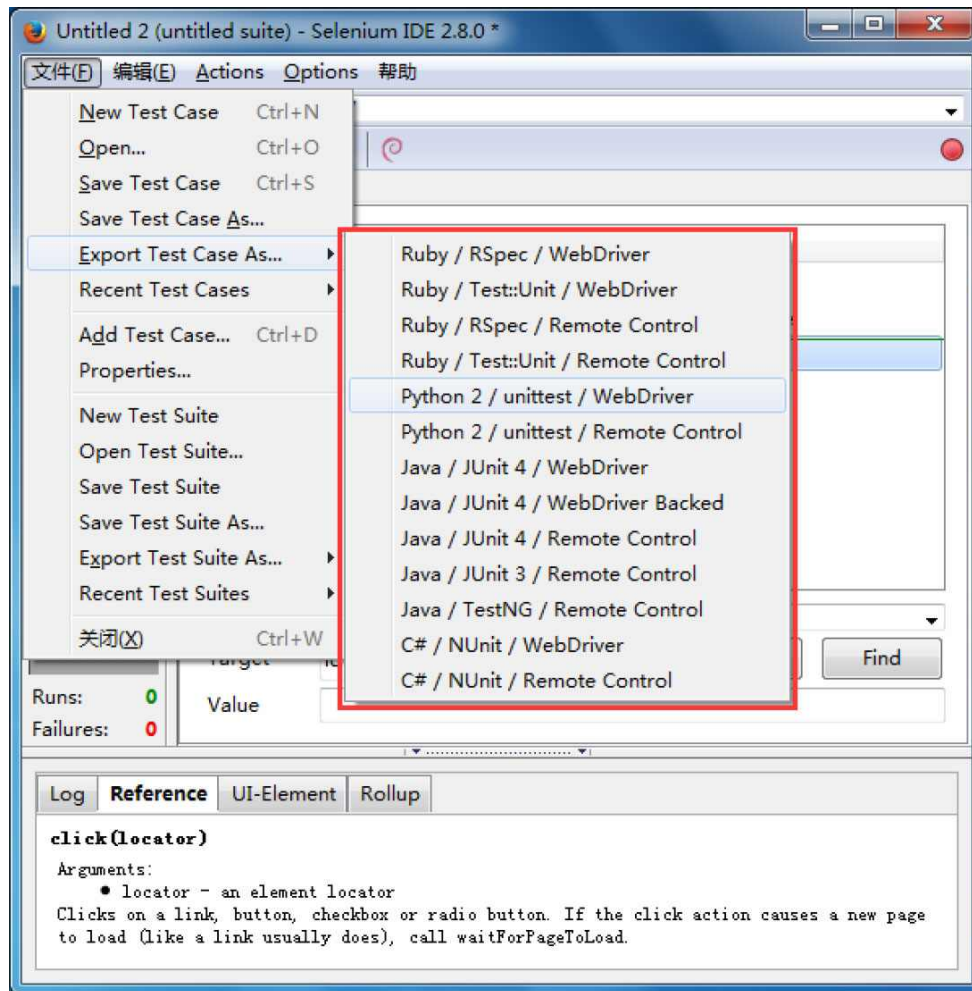
setUp/tearDown 实例级别的setUp/tearDown

```
class Test(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        pass
    def setUp(self):
        pass
    def test(self):
        pass
    def tearDown(self):
        pass
    def tearDownClass(cls):
        pass
```

## 7.3 unittest 单元测试

unittest 是 Python 标准库中用于编写单元测试的框架。它提供了丰富的 API 来支持测试的编写和执行。在 Selenium 中，我们可以使用 unittest 来编写对 Web 应用的测试用例。Selenium IDE 6.0 版本开始支持使用 unittest 来编写测试用例。Selenium IDE 提供了图形化的界面来编写测试用例，并且可以导出为 unittest 格式的测试用例。

在 Selenium IDE 6.0 版本中，我们可以使用 Selenium IDE 提供的图形化界面来编写测试用例。在 Selenium IDE 的“Test Case”窗口中，我们可以选择“Export Test Case As...”选项，将测试用例导出为 unittest 格式的测试用例。Selenium IDE 提供了丰富的 API 来支持测试的编写和执行。在 Selenium IDE 中，我们可以使用 Selenium IDE 提供的图形化界面来编写测试用例。在 Selenium IDE 的“Test Case”窗口中，我们可以选择“Export Test Case As...”选项，将测试用例导出为 unittest 格式的测试用例。



## 7.1 Selenium IDE

### Selenium IDE

- Ruby/RSpec/WebDriver
- Ruby/RSpec/Remote Control
- Ruby/Test::Unit/WebDriver
- Ruby/Test::Unit/Remote Control
- Python2/unittest/WebDriver
- Python2/unittest/Remote Control
- Java/Junit4/WebDriver
- Java/Junit4/WebDriver Backed

- Java/Junit4/Remote Control
- Java/Junit3/Remote Control
- Java/TestNG/Remote Control
- C#/Nunit/WebDriver
- C#/Nunit/Remote Control

Selenium IDESelenium IDE 是一个轻量级的 Selenium 测试工具，它允许用户通过录制和回放的方式，快速编写和执行 Selenium 测试脚本。它支持多种浏览器，并且可以与 Selenium 服务器进行交互。

Python 是一个广泛使用的脚本语言，它支持多种编程范式，包括面向对象、函数式和过程式编程。Python 2 和 Python 3 是两个主要的版本，它们在语法和功能上有一些差异。unittest 是一个内置的单元测试框架，它允许用户编写测试用例并执行它们。WebDriver 是一个接口，它允许用户通过代码来控制浏览器。Python2/unittest/WebDriver 是一个组合，它允许用户在 Python 2 环境中使用 unittest 框架来测试 WebDriver 驱动的浏览器。

Python 2 和 Python 3 是两个主要的版本，它们在语法和功能上有一些差异。Python ILDE 是一个轻量级的 Python 测试工具，它允许用户通过录制和回放的方式，快速编写和执行 Python 测试脚本。它支持多种浏览器，并且可以与 Selenium 服务器进行交互。

baidu.py

```
# -*- coding: utf-8 -*-
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import Select
from selenium.common.exceptions import NoSuchElementException
from selenium.common.exceptions import NoAlertPresentException
import unittest, time, re

class BaiduTest(unittest.TestCase):
    def setUp(self):
```

```

self.driver = webdriver.Firefox()
self.driver.implicitly_wait(30)
self.base_url = "http://www.baidu.com/"
self.verificatiOnErrors = []
self.accept_next_alert = True

def test_baidu(self):
    driver = self.driver
    driver.get(self.base_url + "/")
    driver.find_element_by_id("kw").clear()
        driver.find_element_by_id("kw").send_keys("selenium
ide")

    driver.find_element_by_id("su").click()
def is_element_present(self, how, what):
    try:
        self.driver.find_element(by=how, value=what)
    except NoSuchElementException, e:
        return False
    return True

def is_alert_present(self):
    try:
        self.driver.switch_to_alert()
    except NoAlertPresentException, e:
        return False
    return True

def close_alert_and_get_its_text(self):
    try:
        alert = self.driver.switch_to_alert()

```





```
self.verificationErrors = []
self.accept_next_alert = True
```

setUp()tearDown()  
URL

```
implicitly_wait(30)
```

```
self.verificationErrors = []
```

```
self.accept_next_alert = True
```

```
def test_baidu(self):
    driver = self.driver
    driver.get(self.base_url + "/")
    driver.find_element_by_id("kw").clear()
    driver.find_element_by_id("kw").send_keys("selenium ide")
    driver.find_element_by_id("su").click()
```

test\_baidu

```
def is_element_present(self, how, what):
    try:
        self.driver.find_element(by=how, value=what)
    except NoSuchElementException, e:
        return False
    return True
```

is\_element\_present()find\_element()  
howwhatTrue

False try...except.... Python

```
def is_alert_present(self):
    try:
        self.driver.switch_to_alert()
    except NoAlertPresentException, e:
        return False
    return True
```

is\_alert\_present() WebDriver  
switch\_to\_alert() True  
NoAlertPresentException False

True  
driver.switch\_to\_alert().text  
True False

```
def close_alert_and_get_its_text(self):
    try:
        alert = self.driver.switch_to_alert()
        alert_text = alert.text
        if self.accept_next_alert:
            alert.accept()
        else:
            alert.dismiss()
        return alert_text
    finally:
        self.accept_next_alert = True
```

```
close_alert_and_get_its_text()
switch_to_alert()
text = self.driver.find_element_by_id('alert_text')
if text == 'fail':
    accept_next_alert()
    self.setUp()
    self.assertTrue(True)
    accept()
    self.dismiss()
    self.assertEqual(1, 1)
```

```
def tearDown(self):
    self.driver.quit()
    self.assertEqual([], self.errors)
```

tearDown() 方法在每次测试运行结束后被调用。它用于清理测试环境，例如关闭浏览器驱动。

setUp() 方法在每次测试运行前被调用。它用于设置测试环境，例如启动浏览器驱动。assertEqual() 方法用于断言两个值是否相等。如果断言失败，将引发 AssertionError 异常。

```
if __name__ == '__main__':
    unittest.main()
```

unittest.main() 方法用于运行测试套件。它会自动发现并运行所有测试用例。

## 7.4 Web 测试

Web 测试是指使用 unittest 框架对 Web 应用程序进行测试。它通常涉及使用 Selenium 等工具来模拟用户操作。

```
test_project/
├── test_case/
│   ├── test_baidu.py
│   └── test_youdao.py
├── report/
│   └── login.txt
└── runtest.py
```

Web

test\_baidu.py

```
from selenium import webdriver
import unittest
import time

class MyTest(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Firefox()
        self.driver.maximize_window()
        self.driver.implicitly_wait(10)
        self.base_url = "http://www.baidu.com"

    def test_baidu(self):
```

```

        driver = self.driver
        driver.get(self.base_url + "/")
        driver.find_element_by_id("kw").clear()
        driver.find_element_by_id("kw").send_keys("unittest")
        driver.find_element_by_id("su").click()
        time.sleep(2)
        title = driver.title
        self.assertEqual(title, "unittest_????")
    def tearDown(self):
        self.driver.quit()
if __name__ == "__main__":
    unittest.main()

```

## test\_youdao.py

```

from selenium import webdriver
import unittest
import time
class MyTest(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Firefox()
        self.driver.maximize_window()
        self.driver.implicitly_wait(10)
        self.base_url = "http://www.youdao.com"
    def test_youdao(self):
        driver = self.driver
        driver.get(self.base_url + "/")

```

```

        driver.find_element_by_id("query").clear()

driver.find_element_by_id("query").send_keys("webdriver")
        driver.find_element_by_id("qb").click()
        time.sleep(2)
        title = driver.title
        self.assertEqual(title, "webdriver - 百度")
    def tearDown(self):
        self.driver.close()
if __name__ == "__main__":
    unittest.main()

```

在 test\_case/ 目录下新建 test\_baidu.py 和 test\_youdao.py 两个测试用例，Web 测试用例

runtest.py 7.1.5 运行 runtest.py 测试用例，运行命令如下：
   
 运行命令：“./test\_project/test\_case” 运行 test\_case 测试用例

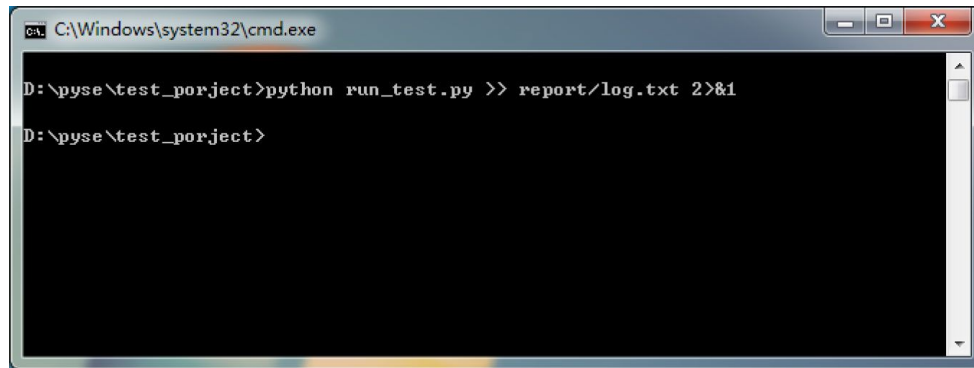
运行结果

运行结果 report 运行结果 report 运行结果 report
   
 运行结果 log.txt 运行结果 dos 运行结果

运行 Windows 运行结果 .../test\_project/运行结果 7.2
   
 运行

cmd.exe

```
> python runtest.py >> report/log.txt 2>&1
```



7.2 runtest.py

...../report/log.txt

log.txt

..

-----

-----

Ran 2 tests in 19.391s

OK

□□□□

Python unittest  
Web  
Web









## Python Shell

```
>>> import HTMLTestRunner
```

```
>>>
```

### 8.1.1 `HTMLTestRunner`

`HTMLTestRunner.py` Python 2 Python 3  
Python 2 Python 3 `HTMLTestRunner.py`

`HTMLTestRu...`

```
# 94
```

```
import StringIO
```

```
import io
```

```
#539
```

```
self.outputBuffer = StringIO.StringIO()
```

```
self.outputBuffer = io.StringIO()
```

```
631
```

```
print >>sys.stderr, '\nTime Elapsed: %s' % (self.stopTime-  
self.startTime)
```

```
print(sys.stderr, '\nTime Elapsed: %s' % (self.stopTime-
```

```
self.startTime))
```

```
#642
```

```
if not rmap.has_key(cls):
```

```
    
```

```
if not cls in rmap:
```

```
766
```

```
uo = o.decode('latin-1')
```

```
    
```

```
uo = e
```

```
772
```

```
ue = e.decode('latin-1')
```

```
    
```

```
ue = e
```

## 8.1.2 HTML

test\_baidu.py HTMLTestRunner

test\_baidu.py

```
from selenium import webdriver
```

```
import unittest
```

```
from HTMLTestRunner import HTMLTestRunner
```

```

class Baidu(unittest.TestCase):

    def setUp(self):
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(10)
        self.base_url = "http://www.baidu.com/"

    def test_baidu_search(self):
        driver = self.driver
        driver.get(self.base_url)

        driver.find_element_by_id("kw").send_keys("HTMLTestRunner")
        driver.find_element_by_id("su").click()

    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":

    testunit = unittest.TestSuite()
    testunit.addTest(Baidu("test_baidu_search"))

    # 生成HTML报告
    fp = open('./result.html', 'wb')
    # 生成报告

```

```
runner = HTMLTestRunner(stream=fp,  
                        title='测试报告',  
                        description='测试报告')
```

```
runner.run(testunit) # 测试  
fp.close() # 关闭文件
```

测试

```
from HTMLTestRunner import HTMLTestRunner
```

```
fp = open('result.html', 'w')  
fp.write('')  
fp.close()
```

```
runner = HTMLTestRunner(stream=fp,  
                        title='测试报告',  
                        description='测试报告')
```

```
runner.run(testunit)  
fp.close()
```

测试报告“result.html”文件生成8.1

| 百度搜索测试报告  |       |      |      |       |                        |
|---|-------|------|------|-------|------------------------|
| Start Time: 2015-07-30 23:17:46   |       |      |      |       |                        |
| Duration: 0:00:08.357000  |       |      |      |       |                        |
| Status: Pass 1  |       |      |      |       |                        |
| 用例执行情况:   |       |      |      |       |                        |
| Show <a href="#">Summary</a> <a href="#">Failed</a> <a href="#">All</a> |       |      |      |       |                        |
| Test Group/Test case  | Count | Pass | Fail | Error | View                   |
| Baidu   | 1     | 1    | 0    | 0     | <a href="#">Detail</a> |
| test_baidu_search   | pass  |      |      |       |                        |
| Total   | 1     | 1    | 0    | 0     |                        |

图8.1 测试结果

## 8.1.3 测试用例的编写

测试用例的编写是测试工作中非常重要的一环。测试用例的编写应该遵循一定的规范，以确保测试用例的可读性和可维护性。在编写测试用例时，应该使用清晰的语言描述测试步骤和预期结果。同时，测试用例应该覆盖所有的测试场景，包括正常情况和异常情况。在编写测试用例时，可以使用一些测试框架提供的功能，如测试用例的分组、测试用例的依赖关系等，以提高测试效率。

在编写测试用例时，可以使用一些测试框架提供的功能，如测试用例的分组、测试用例的依赖关系等，以提高测试效率。同时，测试用例应该覆盖所有的测试场景，包括正常情况和异常情况。在编写测试用例时，可以使用一些测试框架提供的功能，如测试用例的分组、测试用例的依赖关系等，以提高测试效率。

### Python Shell

```
>>> def add(a, b):
    "add()函数用于计算两个数的和"
    return a + b
```

```
>>> add(2, 4)
```

6

```
>>> help(add)
```

```
Help on function add in module __main__:
```

```
add(a, b)
```

```
add() 添加两个数并返回结果
```

```
添加两个数并返回结果
doc string
help() 帮助
```

```
HTMLTestRunner doc string

```

```
baidu.py
```

```
# .....
```

```
class Baidu(unittest.TestCase):
```

```
    ''' 百度测试 '''
```

```
# .....
```

```
def test_baidu_search(self):
```

```
    ''' 百度测试 HTMLTestRunner '''
```

```
# .....
```

```
8.2
```



| 百度搜索测试报告  |       |      |                      |       |                        |
|---|-------|------|----------------------|-------|------------------------|
| Start Time: 2015-10-16 15:08:51   |       |      |                      |       |                        |
| Duration: 0:00:17.379000  |       |      |                      |       |                        |
| Status: Pass 1  |       |      |                      |       |                        |
| 用例执行情况:   |       |      |                      |       |                        |
| Show <a href="#">Summary</a> <a href="#">Failed</a> <a href="#">All</a> |       |      |                      |       |                        |
| Test Group/Test case  | Count | Pass | Fail                 | Error | View                   |
| Baidu: 百度搜索测试   | 1     | 1    | 0                    | 0     | <a href="#">Detail</a> |
| test_baidu_search: 搜索关键字: HTMLTestRunner                                |       |      | <a href="#">pass</a> |       |                        |
| Total   | 1     | 1    | 0                    | 0     |                        |

## 8.2 测试报告

### 8.1.4 测试报告

测试报告是测试人员根据测试用例、测试数据、测试结果等信息，对测试过程进行总结和记录。测试报告是测试人员向项目经理、开发人员、测试人员等提供的重要文档。测试报告可以帮助测试人员了解测试过程，发现问题，改进测试工作。测试报告还可以帮助项目经理、开发人员、测试人员等了解测试进度，评估测试质量，做出决策。

Python time 模块提供了时间相关的功能。time 模块包含以下函数：

Python Shell

```
>>> import time
>>> time.time()
1445694559.2290168

>>> time.ctime()
'Sat Oct 24 21:49:29 2015'
```

```
>>> time.localtime()
time.struct_time(tm_year=2015,      tm_mon=10,      tm_mday=24,
tm_hour=21, tm_min=49,
tm_sec=49, tm_wday=5, tm_yday=297, tm_isdst=0)

>>> time.strftime("%Y_%m_%d %H:%M:%S")
'2015_10_24 21:50:15'
```

time.time() 返回浮点数

time.ctime() 返回字符串

time.localtime() 返回struct\_time结构

time.strftime() 返回字符串

Python 8.1

## 8.1 Python

| 格式符 | 说明               |
|-----|------------------|
| %a  | 本地时间，星期一至五       |
| %A  | 本地时间，星期一至五       |
| %w  | 星期，0表示星期一，6表示星期日 |
| %d  | 本地时间，月/日         |
| %b  | 本地时间，月           |
|     |                  |

|    |                         |
|----|-------------------------|
| %B | □□□□□                   |
| %m | □□□□□□□□                |
| %y | □□□□□□□□□□□□0□99□       |
| %Y | □□□□□□□□□□              |
| %H | 24□□□□□□                |
| %I | 12□□□□□□                |
| %p | □□□AM□PM□□□□□           |
| %M | □□□□□□□□                |
| %S | □□□□□□                  |
| %f | □□□□□□□□□□□□            |
| %Z | □□□□□□□                 |
| %j | □□□□□□□□□□□             |
| %U | □□□□□□□□00□53□□□□□□□□□□ |
| %W | □□□□□□□□00□53□□□□□□□□□□ |
| %x | □□□□□□□□□               |
| %X | □□□□□□□□□               |
| %% | %□□□                    |

□□□□□□□□□□□□□□

## test\_baidu.py

```
import time
```

```
# .....
```

```
if __name__ == "__main__":
```

```
    testunit = unittest.TestSuite()
```

```
    testunit.addTest(Baidu("test_baidu_search"))
```

```
    # 生成测试报告
```

```
    now = time.strftime("%Y-%m-%d %H_%M_%S")
```

```
    # 生成报告文件
```

```
    filename = './' + now + 'result.html'
```

```
    fp = open(filename, 'wb')
```

```
    runner = HTMLTestRunner(stream=fp,
```

```
                            title='测试报告',
```

```
                            description='测试报告')
```

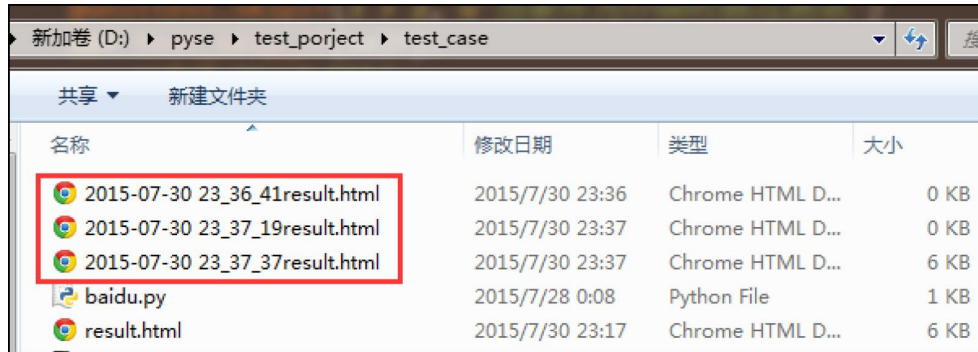
```
    runner.run(testunit)
```

```
    fp.close()
```

```
    时间戳生成报告文件名称now
```

```
now+测试报告8.3
```

111



8.3 4444

### 8.1.5

```

HTMLTestRunner
runtest.py
runtest.py

```

runtest.py

```
import unittest, time

from HTMLTestRunner import HTMLTestRunner

# 测试用例
test_dir = './test_case'

discover = unittest.defaultTestLoader.discover(test_dir, pattern
='test_*.py')

if __name__ == '__main__':
```

```
now = time.strftime("%Y-%m-%d %H_%M_%S")
filename = test_dir + '/' + now + 'result.html'
fp = open(filename, 'wb')
runner = HTMLTestRunner(stream=fp,
                        title='测试报告',
                        description='测试用例执行情况')
runner.run(discover)
fp.close()
```

HTML测试报告8.4

测试报告

Start Time: 2015-07-30 23:55:04

Duration: 0:00:21.158000

Status: Pass 2

用例执行情况:

Show

Summary

Failed

All

| Test Group/Test case          | Count | Pass | Fail | Error | View                   |
|-------------------------------|-------|------|------|-------|------------------------|
| test_baidu.Baidu: 测试百度搜索      | 1     | 1    | 0    | 0     | <a href="#">Detail</a> |
| test_baidu: 搜索关键字: unittest   |       |      |      | pass  |                        |
| test_youdao.Youdao: 测试有道搜索    | 1     | 1    | 0    | 0     | <a href="#">Detail</a> |
| test_youdao: 搜索关键字: webdriver |       |      |      | pass  |                        |
| Total                         | 2     | 2    | 0    | 0     |                        |

8.4 测试报告

# 8.2 测试用例

测试用例是测试人员在测试之前，根据需求规格说明书，设计出的测试通过的条件和测试失败的条件的集合。测试用例是测试人员测试时的重要依据，也是测试人员测试时的重要依据。测试用例是测试人员测试时的重要依据，也是测试人员测试时的重要依据。

SMTP Simple Mail Transfer Protocol  
SMTP is a protocol for sending email messages between servers and from servers to users.

Python's `smtplib` module provides a simple interface to the SMTP protocol. The `SMTP` class in `smtplib` provides the `sendmail()` method to send email messages. For more information, see the `help()` function for the `SMTP` class.

## Python Shell

```
>>> from smtplib import SMTP
```

```
>>> help(SMTP)
```

```
Help on class SMTP in module smtplib:
```

```
| connect(self, host='localhost', port=0)
|     Connect to a host on a given port.
|
|     If the hostname ends with a colon (':') followed by a
number, and
|     there is no port specified, that suffix will be
stripped off and the
|     number interpreted as the port number to use.
```

```
.....
```

```
| login(self, user, password)
|     Log in on an SMTP server that requires authentication.
|
|     The arguments are:
|         - user:      The user name to authenticate with.
```

|           - password: The password for the authentication.

.....

|   quit(self)  
|       Terminate the SMTP session.

.....

|   sendmail(self, from\_addr, to\_addrs, msg, mail\_options=[],  
rcpt\_options=[])

|       This command performs an entire mail transaction.

|

|       The arguments are:

|           - from\_addr       : The address sending this mail.

|           - to\_addrs        : A list of addresses to send this  
mail to. A bare

|                               string will be treated as a list  
with 1 address.

|           - msg             : The message to send.

|           - mail\_options   : List of ESMTP options (such as  
8bitmime) for the

|                               mail command.

|           - rcpt\_options   : List of ESMTP options (such as DSN  
commands) for

|                               all the rcpt commands.

|

|       If there has been no previous EHLO or HELO command this  
session, this



|        method tries ESMTP EHLO first.    If the server does  
ESMTP, message size  
|        and each of the specified options will be passed to it.  
If EHLO  
|        fails, HELO will be tried and ESMTP options suppressed.

.....

SMTP帮助函数help()发送邮件函数sendmail()

connect(host,port)连接SMTP服务器

- host: 要连接的SMTP服务器地址
- port: 要连接的SMTP服务器端口

login(user, password)登录SMTP服务器

- user: 用户名
- password: 密码

sendmail(from\_addr, to\_addrs, msg,...)发送邮件

- from\_addr: 发件人地址
- to\_addrs: 收件人地址列表
- msg: 邮件内容

quit()退出SMTP服务器

Web邮箱mail.126.com  
使用Web邮箱发送邮件的步骤如下：

OutlookFoxmailsmtp.126.com

PythonSMTP

## 8.2.1 HTML

send\_mail.py

```
import smtplib
from email.mime.text import MIMEText
from email.header import Header

# 
smtpserver = 'smtp.sina.com'
# /
user = 'username@sina.com'
password = '123456'
# 
sender = 'username@sina.com'
# 
receiver = 'receive@126.com'
# 
```

```
# HTML
msg = MIMEText('<html><h1><<</h1></html>', 'html', 'utf-8')
msg['Subject'] = Header(subject, 'utf-8')
```

```

SMTPemail
Header()MIMEText()html
receive@126.com8.5

```



## 8.2.2 发送邮件附件

发送邮件附件的完整代码如下：

send\_mail.py

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

# 发件人
smtpserver = 'smtp.sina.com'
# 收件人
sender = 'username@sina.com'
# 收件人
receiver = 'receiver@126.com'
# 发件人/收件人
user = 'username@sina.com'
password = '123456'
# 主题
subject = 'Python send email test'
# 附件
sendfile = open('D:\\testpro\\report\\log.txt', 'rb').read()

att = MIMEText(sendfile, 'base64', 'utf-8')
att["Content-Type"] = 'application/octet-stream'
att["Content-Disposition"] = 'attachment; filename="log.txt"'
```

```

msgRoot = MIMEMultipart('related')
msgRoot['Subject'] = subject
msgRoot.attach(att)

smtp = smtplib.SMTP()
smtp.connect(smtpserver)
smtp.login(user, password)
smtp.sendmail(sender, receiver, msgRoot.as_string())
smtp.quit()

```

下面代码使用MIMEMultipart()函数构造一个多部分邮件8.6



8.6 代码示例

## 8.2.3 发送邮件

下面代码使用Python的smtplib模块发送邮件。该代码将邮件内容存储在log.txt文件中，并将该文件作为附件发送邮件。

find\_file.py

```

import os

# 初始化
result_dir = 'D:\\testpro\\report'

lists = os.listdir(result_dir)

# 按照时间排序
lists.sort(key=lambda fn: os.path.getmtime(result_dir+"\\"+fn))

print(('找到文件 ' + lists[-1]))
file = os.path.join(result_dir, lists[-1])
print(file)

# 使用os.listdir()和os.path.getmtime()对result_dir下的文件进行排序
# 然后使用lists[-1]获取最后一个元素，即最新的文件
# 最后使用os.path.join()将结果目录和文件名组合成完整的路径

```

## Python Shell

```

===== RESTART: D:/test/find_file.py
=====

2015-10-24 22_45_25result.html
D:\testpro\report\2015-10-24 22_45_25result.html

```

## 8.2.4 文件操作



```

msg.as_string())
    smtp.quit()
    print('email has send out !')


# =====
def new_report(testreport):
    lists = os.listdir(testreport)
    lists.sort(key=lambda fn: os.path.getmtime(testreport +
"\\" + fn))
    file_new = os.path.join(testreport, lists[-1])
    print(file_new)
    return file_new


if __name__ == '__main__':

    test_dir = 'D:\\testpro\\test_case'
    test_report = 'D:\\testpro\\report'

    discover = unittest.defaultTestLoader.discover(test_dir,

pattern='test_*.py')
    now = time.strftime("%Y-%m-%d_%H_%M_%S")
    filename = test_report + '\\ ' + now + 'result.html'
    fp = open(filename, 'wb')
    runner = HTMLTestRunner(stream=fp,

```



```

        title='unittest',
        description='unittest报告')

runner.run(discover)
fp.close()

```

```

new_report = new_report(test_report)
send_mail(new_report)    #发送邮件

```

下面我们来对代码进行详细讲解

① `unittest`的`discover()`方法返回一个`HTMLTestRunner`的`run()`方法返回一个`test_report`对象

② `new_report()`方法接收一个`report`对象并返回一个`test_report`对象

③ `send_mail()`方法发送邮件

下面我们来对代码进行详细讲解

| 自动化测试报告 <div>                         发件人: (redacted)@126.com&gt;                         收件人: (无)                         时 间: 2015年08月01日 15:56 (星期六)                     </div> |          |          |          |          |                        |
|--|----------|----------|----------|----------|------------------------|
| <b>测试报告</b>  |          |          |          |          |                        |
| <b>Start Time:</b> 2015-08-01 15:56:06   |          |          |          |          |                        |
| <b>Duration:</b> 0:00:22.637000  |          |          |          |          |                        |
| <b>Status:</b> Pass 2  |          |          |          |          |                        |
| 用例执行情况:  |          |          |          |          |                        |
| Show <a href="#">Summary</a> <a href="#">Failed</a> <a href="#">All</a>  |          |          |          |          |                        |
| Test Group/Test case   | Count    | Pass     | Fail     | Error    | View                   |
| test_baidu.Baidu: 测试百度搜索   | 1        | 1        | 0        | 0        | <a href="#">Detail</a> |
| test_youdao.Youdao: 测试有道搜索   | 1        | 1        | 0        | 0        | <a href="#">Detail</a> |
| <b>Total</b>   | <b>2</b> | <b>2</b> | <b>0</b> | <b>0</b> |                        |

Web Web  
HTML UI page  
HTML API HTML  
Page Object 8.8



126Page Object

po\_model.py

```
from selenium import webdriver
```

```
from selenium.webdriver.common.by import By
```

```
from time import sleep
```

```
class Page(object):
```

```
    '''
```

```
    '''
```

```
    '''
```

```
login_url = 'http://www.126.com'
```

```
def __init__(self, selenium_driver, base_url=login_url):
```

```
    self.base_url = base_url
```

```
    self.driver = selenium_driver
```

```
    self.timeout = 30
```

```
def on_page(self):
```

```
    return self.driver.current_url == (self.base_url + self.url)
```

```
def _open(self, url):
```

```
    url = self.base_url + url
```

```

        self.driver.get(url)
        assert self.on_page(), 'Did not land on %s' % url

    def open(self):
        self._open(self.url)

    def find_element(self, *loc):
        return self.driver.find_element(*loc)


class LoginPage(Page):
    ...

    126
    ...

    url = '/'

    #
    username_loc = (By.ID, "idInput")
    password_loc = (By.ID, "pwdInput")
    submit_loc = (By.ID, "loginBtn")

    # Action
    def type_username(self, username):

self.find_element(*self.username_loc).send_keys(username)

```

```

def type_password(self, password):

self.find_element(*self.password_loc).send_keys(password)


def submit(self):
    self.find_element(*self.submit_loc).click()


def test_user_login(driver, username, password):
    """
    用户名/密码登录
    """
    login_page = LoginPage(driver)
    login_page.open()
    login_page.type_username(username)
    login_page.type_password(password)
    login_page.submit()


def main():
    try:
        driver = webdriver.Firefox()
        username = 'username'
        password = '123456'
        test_user_login(driver, username, password)
        sleep(3)

```

text =

```

driver.find_element_by_xpath("//span[@id='spnUid']").text
    assert(text == 'username@126.com'), "用户名或密码错误!"
finally:
    # 关闭浏览器
    driver.close()

```

```

if __name__ == '__main__':
    main()

```

Page Object模型

## 1. Page Object

po\_model.py

```

# .....
class Page(object):
    '''
    Page Object Model
    '''

    login_url = 'http://www.126.com'

    def __init__(self, selenium_driver, base_url=login_url):
        self.base_url = base_url

```

```

        self.driver = selenium_driver
        self.timeout = 30

    def on_page(self):
        return self.driver.current_url == (self.base_url +
self.url)

    def _open(self, url):
        url = self.base_url + url
        self.driver.get(url)
        assert self.on_page(), 'Did not land on %s' % url

    def open(self):
        self._open(self.url)

    def find_element(self, *loc):
        return self.driver.find_element(*loc)
# .....

```

```

class Page:
    def __init__(self, driver, URL,
base_url, timeout):

```

```

        self.open(URL)
        self._open(URL)
        self.on_page()
        self.find_element()

```

## 2 LoginPage



Page 클래스를 상속받아 LoginPage  
클래스를 생성합니다. Page Object 패턴을 사용합니다.

po\_model.py

# .....

class LoginPage(Page):

...

126

...

url = '/'

# ...

username\_loc = (By.ID, "idInput")

password\_loc = (By.ID, "pwdInput")

submit\_loc = (By.ID, "loginBtn")

# Action

def type\_username(self, username):

self.find\_element(\*self.username\_loc).send\_keys(username)

def type\_password(self, password):

self.find\_element(\*self.password\_loc).send\_keys(password)

```

    def submit(self):
        self.find_element(*self.submit_loc).click()
# .....

LoginPage()
# .....
LoginPage()
# .....

```

### 3 test\_user\_login()

```

po_model.py

.....
def test_user_login(driver, username, password):
    """
    测试用户登录/注册功能
    """
    login_page = LoginPage(driver)
    login_page.open()
    login_page.type_username(username)
    login_page.type_password(password)
    login_page.submit()
.....

test_user_login()
# .....
test_user_login(driver, username, password)
# .....

```

## 4 main()

po\_model.py

# .....

```
def main():
```

```
try:
```

```
driver = webdriver.Firefox()
```

```
username = 'username'
```

```
password = '123456'
```

```
test_user_login(driver, username, password)
```

```
sleep(3)
```

text =

```
driver.find_element_by_xpath("//span[@id='spnUid']").text
```

```
assert(text == 'username@126.com'), "用户名或密码错误!"
```

finally:

# 

|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|

```
driver.close()
```

```
if __name__ == '__main__':
```

```
main()
```

```
main()
```

[illegible]

Page Object Model (POM) is a design pattern used in test automation to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable.

Page Object Model (POM) is a design pattern used in test automation to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable. API Page Object Model (API POM) is a variation of POM used for testing APIs. It focuses on the API endpoints and their responses.

Page Object Model (POM) is a design pattern used in test automation to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable. Page Object Model (POM) is a design pattern used in test automation to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable.

Page Object Model (POM) is a design pattern used in test automation to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable. 10 Page Object Model (POM) is a design pattern used in test automation to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable.

Page Object Model (POM) is a design pattern used in test automation to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable. ATDD (Acceptance Test Driven Development) and BDD (Behavior Driven Development) are testing methodologies that use Page Object Model (POM) to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable.

Page Object Model (POM) is a design pattern used in test automation to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable. Page Object Model (POM) is a design pattern used in test automation to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable.

Page Object Model (POM) is a design pattern used in test automation to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable.

Page Object Model (POM) is a design pattern used in test automation to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable. HTMLTestRunner is a test runner used in test automation to execute test scripts. Page Object Model (POM) is a design pattern used in test automation to organize test data and test logic into reusable components. It helps in maintaining the test scripts and makes them more readable and maintainable.



## 第9章 Selenium Grid2

Selenium WebDriver Selenium IDE Selenium Grid Selenium Grid hub node Selenium Grid Selenium 2 Selenium Grid2 Selenium 2

### Selenium Grid

Selenium Grid Grid Grid1 Grid2 Selenium Selenium Grid2 Selenium 2 Selenium Grid2 Selenium 2

Grid Grid2 Selenium 1 Selenium 2

Grid2 Selenium Server Selenium Server Grid2

### 9.1 Selenium Server

Selenium Server

① Selenium Server

<http://www.seleniumhq.org/download/>

需要安装Selenium Standalone Server  
Download version 2.48.2 selenium-server-standalone-xxx.jar  
jar Java jar Java

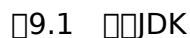
## ② Java

Java  
[http://www.java.com/zh\\_CN/download/manual.jsp](http://www.java.com/zh_CN/download/manual.jsp)

Java

JDKJREJDKJava Development KitJDK  
SDKJavaJREJava Runtime EnvironmentJavaJava

WindowsJDK  
JDK C:\Program Files\Java\jdk1.7.0\_60\9.1



```
□□□□JAVA_HOME
```

```

C:\Program Files\Java\jdk1.7.0_45\

```

CALSS\_PATH

```
%%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;
```

“path” →

path



```
%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin;
```

Windows에서 Java 실행

cmd.exe

```
C:\Users\fnngj> java
```

```
java [-options] class [args...]
```

(옵션)

```
java [-options] -jar jarfile [args...]
```

(jar 파일)

옵션:

-d32 32 비트 (32비트)

-d64 64 비트 (64비트)

-server "server" VM

-hotspot "server" VM 실행 (실행)

VM server.

.....

```
C:\Users\fnngj> javac
```

```
javac <options> <source files>
```

옵션, 옵션:

-g 디버깅 정보

-g:none 디버깅 정보 없음

-g:{lines,vars,source} 디버깅 정보

-nowarn 경고 없음

-verbose 디버깅 정보 출력

-deprecation API 사용 경고

-classpath <[]> []

.....

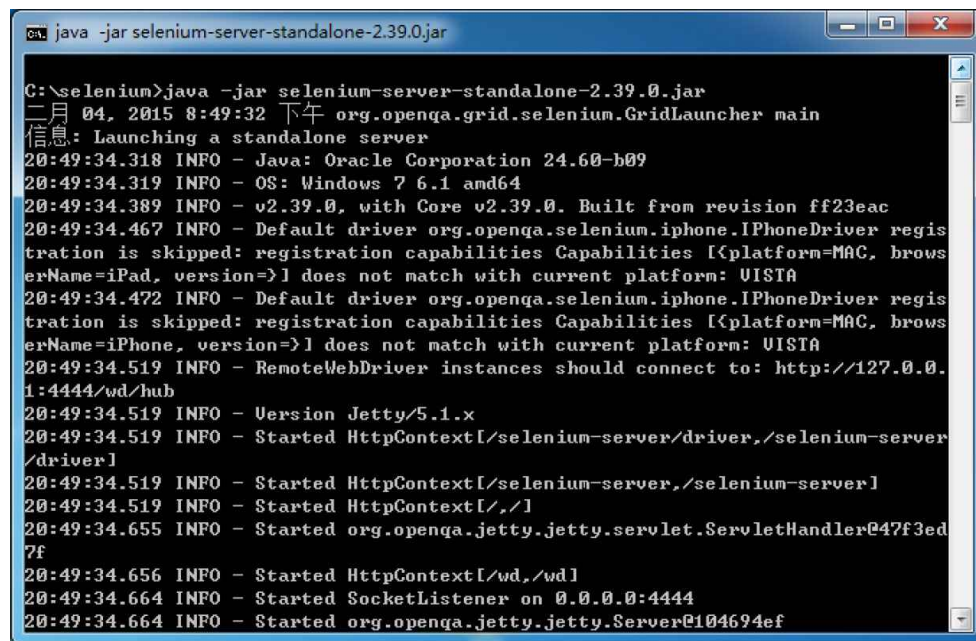
“java”[]class[]

“javac”[]Java[]class[]

### ③ []Selenium Server[]

[]“java”[]Selenium Server[]Selenium Server[]  
[]9.2[]Windows[]Linux[]Selenium  
Server[]

```
> java -jar selenium-server-standalone-2.47.0.jar
```



```
C:\selenium>java -jar selenium-server-standalone-2.39.0.jar
二月 04, 2015 8:49:32 下午 org.openqa.grid.selenium.GridLauncher main
信息: Launching a standalone server
20:49:34.318 INFO - Java: Oracle Corporation 24.60-b09
20:49:34.319 INFO - OS: Windows 7 6.1 amd64
20:49:34.389 INFO - v2.39.0, with Core v2.39.0. Built from revision ff23eac
20:49:34.467 INFO - Default driver org.openqa.selenium.iphone.IPhoneDriver registration is skipped: registration capabilities Capabilities [{platform=MAC, browserName=iPad, version=} does not match with current platform: VISTA
20:49:34.472 INFO - Default driver org.openqa.selenium.iphone.IPhoneDriver registration is skipped: registration capabilities Capabilities [{platform=MAC, browserName=iPhone, version=} does not match with current platform: VISTA
20:49:34.519 INFO - RemoteWebDriver instances should connect to: http://127.0.0.1:4444/wd/hub
20:49:34.519 INFO - Version Jetty/5.1.x
20:49:34.519 INFO - Started HttpContext[/selenium-server/driver,/selenium-server/driver]
20:49:34.519 INFO - Started HttpContext[/selenium-server,/selenium-server]
20:49:34.519 INFO - Started HttpContext[/,/]
20:49:34.655 INFO - Started org.openqa.jetty.jetty.servlet.ServletHandler@47f3ed7f
20:49:34.656 INFO - Started HttpContext[/wd,/wd]
20:49:34.664 INFO - Started SocketListener on 0.0.0.0:4444
20:49:34.664 INFO - Started org.openqa.jetty.jetty.Server@104694ef
```

### 9.2 []Selenium Server

[]1[]Selenium1.0 RC[]Selenium Server[]  
[]RC[]Web Driver[]

sel\_rc.py

```
from selenium import selenium
```

```
sel = selenium("localhost", 4444, "*firefox",  
"http://www.baidu.com/")
```

```
sel.start()
```

```
sel.open("/")
```

```
sel.type("id=q", "selenium grid")
```

```
sel.click("id=btn")
```

```
sel.wait_for_page_to_load("30000")
```

```
sel.stop()
```

## 9.2 Selenium Grid

Grid是由一个或多个hub和多个node组成的。hub是客户端，node是服务器。hub负责接收客户端的请求，并将请求分发给相应的node处理。node处理完请求后，将结果返回给hub，再由hub返回给客户端。Grid支持分布式测试，可以同时运行多个测试用例，提高测试效率。

```
> java -jar selenium-server-standalone-x.xx.x.jar -role hub
```

```
> java -jar selenium-server-standalone-x.xx.x.jar -role node
```

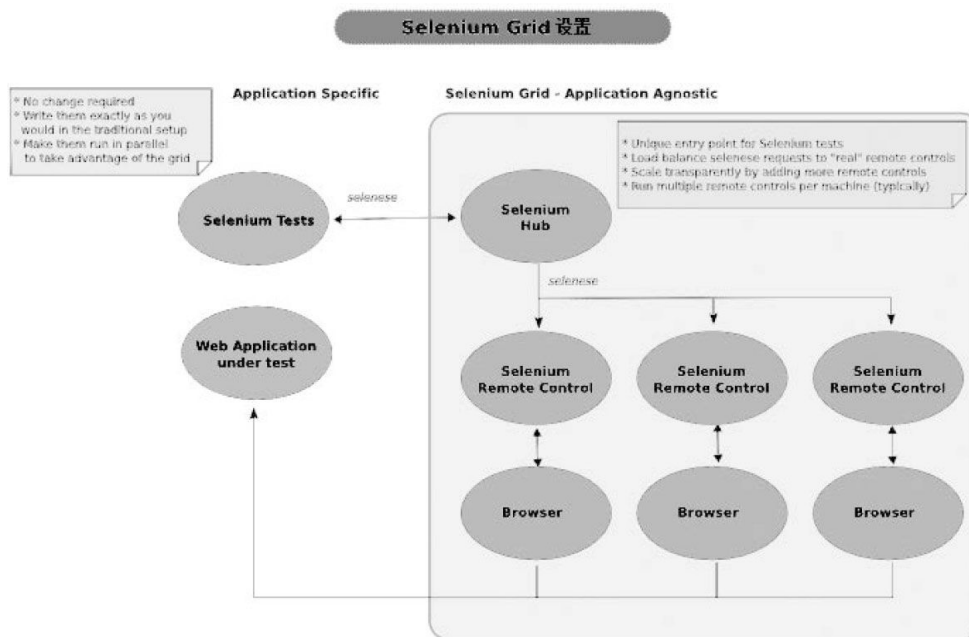
hub node hub 4444 node  
5555 node node node node node node  
node

```
> java -jar selenium-server-standalone-x.xx.x.jar -role node -port 5555
```

```
> java -jar selenium-server-standalone-x.xx.x.jar -role node -port 5556
```

```
> java -jar selenium-server-standalone-x.xx.x.jar -role node -port 5557
```

## Grid 9.3

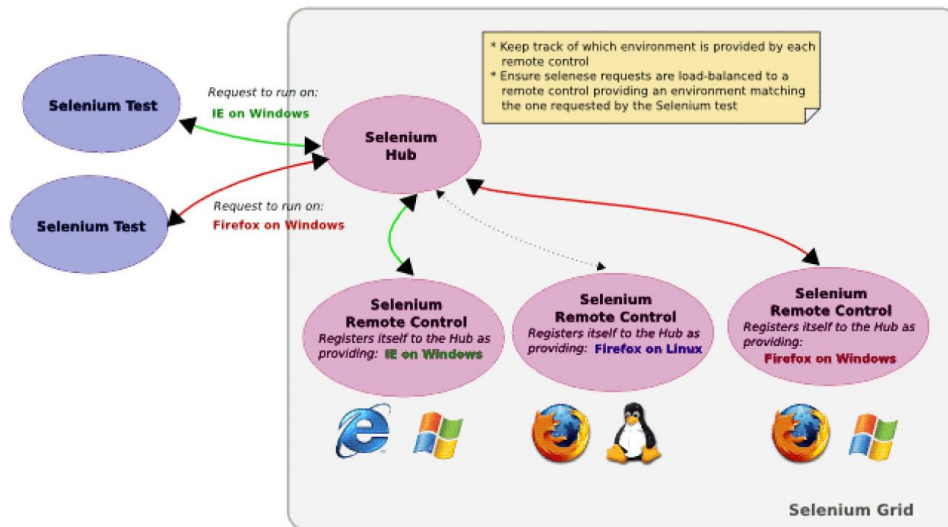


9.3 Selenium Grid

Grid 11 Python Grid

LinuxFirefox  
GridLinuxFirefox  
9.4

#### Selenium Grid : Requesting a Specific Environment



9.4

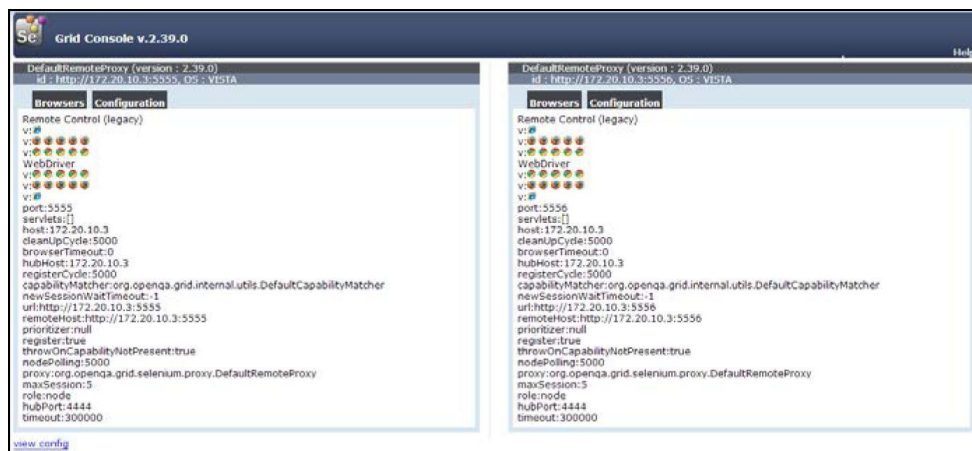
hubnode9.5

```
cmd java -jar selenium-server-standalone-2.39.0.jar -role hub
E:\selenium>java -jar selenium-server-standalone-2.39.0.jar -role hub
cmd java -jar selenium-server-standalone-2.39.0.jar -role node -port 5555
E:\selenium>java -jar selenium-server-standalone-2.39.0.jar -role node -port 5555
cmd java -jar selenium-server-standalone-2.39.0.jar -role node -port 5556
E:\selenium>java -jar selenium-server-standalone-2.39.0.jar -role node -port 5556
二月 12, 2015 10:08:08 上午 org.openqa.grid.selenium.GridLauncher main
信息: Launching a selenium grid node
10:08:14.301 INFO - Java: Oracle Corporation 24.60-b09
10:08:14.301 INFO - OS: Windows 7 6.1 amd64
10:08:14.317 INFO - v2.39.0, with Core v2.39.0. Built from revision ff23eac
10:08:14.395 INFO - Default driver org.openqa.selenium.iphone.IPhoneDriver registration is skipped: registration capabilities Capabilities [{platform=MAC, browserName=iPhone, version=} does not match with current platform: VISTA
10:08:14.395 INFO - Default driver org.openqa.selenium.iphone.IPhoneDriver registration is skipped: registration capabilities Capabilities [{platform=MAC, browserName=iPad, version=} does not match with current platform: VISTA
10:08:14.426 INFO - RemoteWebDriver instances should connect to: http://127.0.0.1:5556/wd/hub
10:08:14.426 INFO - Version Jetty/5.1.x
10:08:14.426 INFO - Started HttpContext[/selenium-server/driver,/selenium-server/driver]
10:08:14.426 INFO - Started HttpContext[/selenium-server,/selenium-server]
10:08:14.426 INFO - Started HttpContext[/,/]
10:08:14.441 INFO - Started org.openqa.jetty.jetty.servlet.ServletHandler@4bac5211
10:08:14.441 INFO - Started HttpContext[/wd,/wd]
10:08:14.441 INFO - Started SocketListener on 0.0.0.0:5556
```

## 9.5 hub node

Grid http://127.0.0.1:4444/grid/console

## 9.6



## 9.6 node

node(5555)

Remote Control (legacy)

v: 

v:     

v:     

WebDriver

v:     

v:     

v: 

port:5555

servlets:[]

host:172.20.10.3

cleanUpCycle:5000

browserTimeout:0

hubHost:172.20.10.3

```
registerCycle:5000
capabilityMatcher:org.openqa.grid.internal.utils.DefaultCapability
Matcher
newSessionWaitTimeout:-1
url:http://172.20.10.3:5555
remoteHost:http://172.20.10.3:5555
prioritizer:null
register:true
throwOnCapabilityNotPresent:true
nodePolling:5000
proxy:org.openqa.grid.selenium.proxy.DefaultRemoteProxy
maxSession:5
role:node
hubPort:4444
timeout:300000
```

## 9.3 Remote

RemoteWebDriver is a Selenium WebDriver implementation that runs on a remote Selenium WebDriver server. It is used to interact with a remote Selenium WebDriver server.

```
driver = webdriver.Firefox()
```

```
driver = webdriver.Chrome()
```

```
driver = webdriver.Ie()
```



### 9.3.1 WebDriver

| 名称           | 修改日期            | 类型                | 大小   |
|--------------|-----------------|-------------------|------|
| android      | 2015/5/24 21:19 | 文件夹               |      |
| chrome       | 2015/5/24 21:19 | 文件夹               |      |
| common       | 2015/5/24 21:19 | 文件夹               |      |
| firefox      | 2015/5/24 21:19 | 文件夹               |      |
| ie           | 2015/5/24 21:19 | 文件夹               |      |
| opera        | 2015/5/24 21:19 | 文件夹               |      |
| phantomjs    | 2015/5/24 21:19 | 文件夹               |      |
| remote       | 2015/5/24 21:19 | 文件夹               |      |
| safari       | 2015/5/24 21:19 | 文件夹               |      |
| support      | 2015/5/24 21:19 | 文件夹               |      |
| __init__.py  | 2015/5/24 21:19 | Python File       | 2 KB |
| __init__.pyc | 2015/5/24 21:19 | Compiled Pytho... | 2 KB |

WebDriver.py Firefox Chrome IE remote Selenium Server

webdriver.py

```

.....
from .firefox_binary import FirefoxBinary
from selenium.webdriver.firefox.firefox_profile import
FirefoxProfile
from selenium.webdriver.remote.webdriver import WebDriver as
RemoteWebDriver
.....

class WebDriver(RemoteWebDriver):

    # There is no native event support on Mac
    NATIVE_EVENTS_ALLOWED = sys.platform != "darwin"

    def __init__(self, firefox_profile=None, firefox_binary=None,
timeout=30,
capabilities=None, executable_path='wires'):

        self.binary = firefox_binary
        self.profile = firefox_profile
.....

    def __init__(self, SeleniumFirefox,
firefox_profile,firefox_binary,
firefox_binary.py,firefox_profile.py,
):

        SeleniumFirefox(selenium.webdriver.Firefox())
        ..../selenium/webdriver/firefox/webdriver.py
        WebDriver../selenium/webdriver/__init__.py

```

\_\_init\_\_.py

.....

```
from .firefox.webdriver import WebDriver as Firefox
from .firefox.firefox_profile import FirefoxProfile
from .chrome.webdriver import WebDriver as Chrome
from .chrome.options import Options as ChromeOptions
from .ie.webdriver import WebDriver as Ie
from .edge.webdriver import WebDriver as Edge
from .opera.webdriver import WebDriver as Opera
from .safari.webdriver import WebDriver as Safari
from .blackberry.webdriver import WebDriver as BlackBerry
from .phantomjs.webdriver import WebDriver as PhantomJS
from .android.webdriver import WebDriver as Android
from .remote.webdriver import WebDriver as Remote
from .common.desired_capabilities import DesiredCapabilities
from .common.action_chains import ActionChains
from .common.touch_actions import TouchActions
from .common.proxy import Proxy
```

\_\_version\_\_ = '2.47.0'

WebDriver  
Firefox Chrome IE

selenium webdriver/chrome Chrome  
webdriver.py

webdriver.py

.....

```
from selenium.webdriver.remote.webdriver import WebDriver as  
RemoteWebDriver
```

.....

```
class WebDriver(RemoteWebDriver):
```

```
    """
```

Controls the ChromeDriver and allows you to drive the browser.

You will need to download the ChromeDriver executable from  
<http://chromedriver.storage.googleapis.com/index.html>

```
    """
```

```
def __init__(self, executable_path="chromedriver", port=0,  
             chrome_options=None, service_args=None,  
             desired_capabilities=None, service_log_path=None):
```

```
    """
```

Creates a new instance of the chrome driver.

Starts the service and then creates new instance of chrome  
driver.

:Args:

- executable\_path - path to the executable. If the  
default is used it

assumes the executable is in the \$PATH

- port - port you would like the service to run, if left

as 0, a free  
port will be found.

- desired\_capabilities: Dictionary object with non-browser specific capabilities only, such as "proxy" or "loggingPref".
- chrome\_options: this takes an instance of ChromeOptions

```
"""
```

.....

```
class WebDriver(object):
    """Selenium WebDriver
    chromedriver.exe executable_path chromedriver
    """
```

```
Firefox Chrome WebDriver
RemoteWebDriver remote WebDriver
WebDriver?
```

```
selenium.webdriver.remote.webdriver.py
```

```
webdriver.py
```

.....

```
class WebDriver(object):
```

```
"""
```

Controls a browser by sending commands to a remote server.

This server is expected to be running the WebDriver wire  
protocol as defined

here: <http://code.google.com/p/selenium/wiki/JsonWireProtocol>

:Attributes:

- `command_executor` - The `command.CommandExecutor` object used to execute commands.

- `error_handler` - `errorhandler.ErrorHandler` object used to verify that the server did not return an error.

- `session_id` - The session ID to send with every command.

- `capabilities` - A dictionary of capabilities of the underlying browser for this instance's session.

- `proxy` - A `selenium.webdriver.common.proxy.Proxy` object, to specify a proxy for the browser to use.

"""

```
def __init__(self,
command_executor='http://127.0.0.1:4444/wd/hub',
desired_capabilities=None, browser_profile=None,
proxy=None,
keep_alive=False):
```

"""

Create a new driver that will issue commands using the wire protocol.

:Args:

- `command_executor` - Either a `command.CommandExecutor` object or a string

that specifies the URL of a remote server to send commands to.

- desired\_capabilities - Dictionary holding predefined values for

starting a browser

- browser\_profile - A

selenium.webdriver.firefox.firefox\_profile.FirefoxProfile object.

Only used

if Firefox is requested.

"""

.....

```
WebDriver __init__()
command_executor127.0.0.14444
```

```
desired_capabilities
selenium webdriver/common
desired_capabilities.py
```

desired\_ca...

.....

```
class DesiredCapabilities(object):
```

.....

```
FIREFOX = {
    "browserName": "firefox",
```

```

    "version": "",
    "platform": "ANY",
    "javascriptEnabled": True,
    "marionette": False,
}

```

.....

```

CHROME = {
    "browserName": "chrome",
    "version": "",
    "platform": "ANY",
    "javascriptEnabled": True,
}

```

.....

'browserName': 'chrome'    chrome(Firefox)

'version': ''   

'platform': 'ANY'    ANY

'javascriptEnabled': True    JavaScript

"marionette": False    marionette Python

gecko marionette Firefox Firefox OS  
Firefox

DesiredCapabilities



## **FIREFOX**

```
= {"browser Name": "firefox", "version": "", "plat form":  
"ANY", "javascriptEnabled": True, "marionette": False, }
```

## **INTERNETEXPLORER**

```
= {"browserName": "internet explorer", "version": "", "platform":  
"WINDOWS", "javascriptEnabled": True, }
```

## **EDGE**

```
= {"browserName": "MicrosoftEdge", "version": "", "platform":  
"WINDOWS" }
```

## **CHROME**

```
= {"browserName": "chrome", "version": "", "platform": "ANY",  
"javascriptEnabled": True, }
```

## **OPERA**

```
= {"browserName": "opera", "version": "", "platform": "ANY",  
"jvas criptEnabled": True, }
```

## **SAFARI**

```
= {"browserName": "safari", "ve rsion": "", "pla tform": "ANY",
```

```
"javascriptEnabled": True, }
```

### **HTMLUNIT**

```
= {"browserName": "htmlunit", "version": "", "platform": "ANY", }
```

### **HTMLUNITWITHJS**

```
= {"browserName": "htmlunit", "version": "firefox", "platform":  
"ANY", "javascriptEnabled": True, }
```

### **IPHONE**

```
= {"browserName": "iPhone", "version": "", "platform": "MAC",  
"javascriptEnabled": True, }
```

### **IPAD**

```
= {"browserName": "iPad", "version": "", "platform": "MAC",  
"javascriptEnabled": True, }
```

### **ANDROID**

```
= {"browserName": "android", "version": "", "platform":  
"ANDROID", "javascriptEnabled": True, }
```

### **PHANTOMJS**



```
driver.get('http://www.baidu.com')
driver.find_element_by_id("kw").send_keys("remote")
driver.find_element_by_id("su").click()

driver.quit()
```

WebDriver.Remote() 是 webdriver.Chrome() 的父类  
Remote() 是 webdriver.Remote() 的父类

## 9.3.3 搭建 Selenium 环境

Selenium Server 是 Selenium 的远程服务器，Remote 是 Selenium 的客户端，  
Selenium Server 和 Remote 之间通过 WebDriver 协议进行通信。

搭建 Selenium 环境需要安装 hub 和 node。

```
> java -jar selenium-server-standalone-2.47.0.jar -role hub

> java -jar selenium-server-standalone-2.47.0.jar -role node -
port 5555

> java -jar selenium-server-standalone-2.47.0.jar -role node -
port 5556
```

搭建 Selenium 环境需要安装 hub 和 node。

remote\_ts.py

```
from selenium.webdriver import Remote
```

```
# 测试用例
```

```
lists = {'http://127.0.0.1:4444/wd/hub': 'chrome',  
         'http://127.0.0.1:5555/wd/hub': 'firefox',  
         'http://127.0.0.1:5556/wd/hub': 'internet explorer'}
```

```
# 遍历测试用例
```

```
for host, browser in lists.items():  
    print(host, browser)  
    driver = Remote(command_executor=host,  
                    desired_capabilities={'platform': 'ANY',  
                                         'browserName': browser,  
                                         'version': '',  
                                         'javascriptEnabled':
```

```
True
```

```
}
```

```
)
```

```
driver.get("http://www.baidu.com")  
driver.find_element_by_id("kw").send_keys(browser)  
driver.find_element_by_id("su").click()  
driver.close()
```

```
遍历lists中的每个IP地址for每个lists中的  
Remote()方法
```

# 1. node

搭建hub和node  
搭建

- 搭建hub和node并ping测试
- 搭建node并配置path
- 搭建Java Selenium Server

## 2. 搭建

① 搭建hub IP 172.16.10.66

```
> java -jar selenium-server-standalone-2.47.0.jar -role hub
```

② 搭建node Ubuntu IP 172.16.10.34

```
$ java -jar selenium-server-standalone-2.47.0.jar -role node -  
port 5555 -hub  
http://172.16.10.66:4444/grid/register
```

5555 hub IP 172.16.10.66

③ 搭建IP 172.16.10.34 Firefox

remote\_ts.py

.....

# 搭建

```
lists = {'http://127.0.0.1:4444/wd/hub': 'chrome',  
         'http://127.0.0.1:5555/wd/hub': 'internet explorer',
```

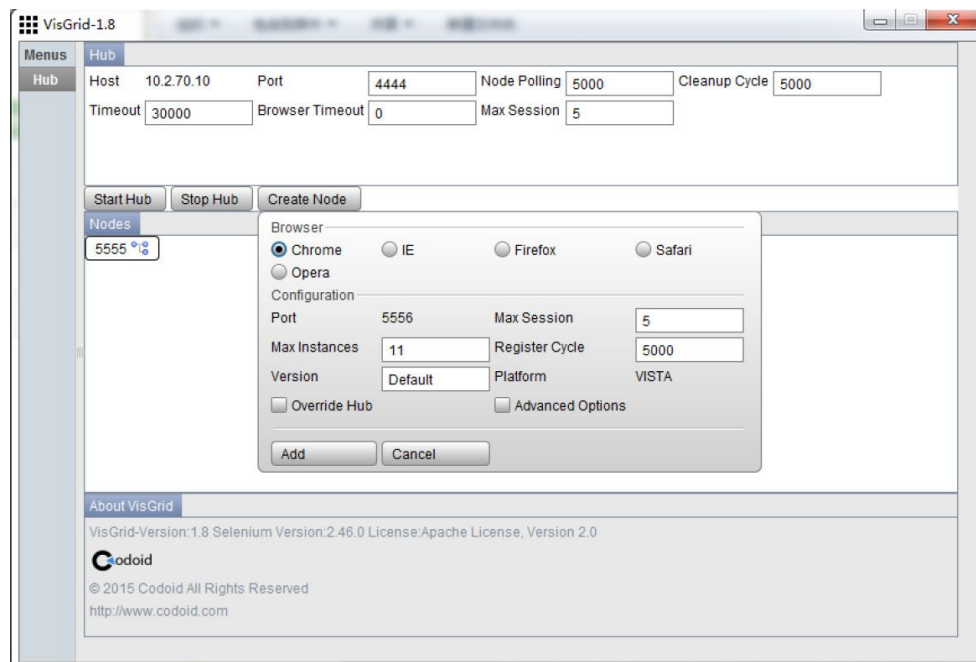
■■■■■

111

```

Selenium Server startup.bat
VisGrid 9.8

```



## 9.4 WebDriver

9.3 WebDriver Firefox Driver Chrome Driver IEDriverServer WebDriver

WebDriver/9.1

### 9.1 WebDriver

| Android    |                    | Android<br>WebView                         |
|------------|--------------------|--|
| BlackBerry |                    |  |
| Firefox    | Selenium           | Selenium(WebDriver)<br>Selenium<br>Firefox |
| Chrome     | chromedriver.exe   | WebDriver<br>Selenium<br>Chrome            |
| IE         | IEDriverServer.exe | IE   |



|           |                        |   |
|-----------|------------------------|---|
| Edge      | MicrosoftWebDriver.exe | Windows10<br>Edge   |
| Opera     | operadriver.exe        | Opera<br>OperaChromiumDriver<br>OperaDriver<br>ChromeDriver |
| Safari    | Selenium Server        | Safari<br>MAC<br>Windows                                    |
| HtmlUnit  | Selenium Server        | HtmlUnit<br>HTML<br>Selenium Server                         |
| PhantomJS | phantomjs.exe          | PhantomJS<br>JavaScript A PI<br>WebKit<br>HtmlUnit          |

## 1

WebDriverAndroidBlackBerry  
Android  
AppiumAppiumWebDriveriOSAndroid  
Web

BlackBerry

## 2

WebDriver 支援するブラウザは Firefox、Chrome、IE、Edge、Opera、Safari

ブラウザのバージョンは最新にしてください

### 3. ブラウザ

HtmlUnit、PhantomJS はブラウザの動作をシミュレートするライブラリです。html、Java Script、CSS、GUI などの操作が可能です。

ブラウザの種類

ブラウザの種類は “Rendering Engine” と呼ばれる “エンジン” によって決まります。 “エンジン” は HTML、JavaScript、CSS などの操作を行います。

#### Trident

Trident は IE のエンジンで、1997 年に IE 4 で登場しました。Mosaic は最初の実装で、IE 11 以降は “IE” と呼ばれる Trident のエンジンで動作します。

Trident は IE のエンジンで、1997 年に IE 4 で登場しました。Mosaic は最初の実装で、IE 11 以降は “IE” と呼ばれる Trident のエンジンで動作します。

#### Gecko

GeckoFirefoxFirefoxNetscape 6Mozilla Firefox  
GeckoGecko

## Presto

PrestoOpera2003Opera 7  
Opera 12.17Opera  
OperaOperaGoogle ChromeBlink

## Webkit

WebkitSafariChrome  
Safari2005Webkit  
WebkitChromeChrome  
Webkit3360

## Blink

BlinkGoogleOpera Software  
WebKitWebCoreChrome28Opera15  
Yandex

## 9.4.1 Edge

Selenium 2.47.0EdgeEdge  
Windows 10IE

#####Selenium#####  
#####

## Python Shell

```
>>> from selenium import webdriver
```

```
>>> driver = webdriver.Edge()
```

Traceback (most recent call last):

```
File "C:\Python35\lib\site-  
packages\selenium\webdriver\edge\service.py",  
line 54, in start
```

```
    stdout=PIPE, stderr=PIPE)
```

```
File "C:\Python35\lib\subprocess.py", line 950, in __init__  
    restore_signals, start_new_session)
```

```
File "C:\Python35\lib\subprocess.py", line 1220, in  
_execute_child  
    startupinfo)
```

FileNotFoundError: [WinError 2] #####

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

```
File "<pyshell#1>", line 1, in <module>
```

```
    driver = webdriver.Edge()
```

```
File "C:\Python35\lib\site-  
packages\selenium\webdriver\edge\webdriver.py",  
line 34, in __init__
```

```
self.edge_service.start()
File "C:\Python35\lib\site-
packages\selenium\webdriver\edge\service.py",
line 59, in start
    "The EdgeDriver executable needs to be available in the path.
"
```

```
selenium.common.exceptions.WebDriverException: Message: The
EdgeDriver
executable needs to be available in the path. Please download from
http://go.microsoft.com/fwlink/?LinkId=619687
```

```
EdgeMicrosoftWebDriver.msi
path
MicrosoftWebDriver.exeC:\Python35
path
```

```
Edge
```

```
baidu.py
```

```
from selenium import webdriver
```

```
driver = webdriver.Edge()
driver.get("http://www.baidu.com")
```

```
driver.find_element_by_id("kw").send_keys("Edge")
driver.find_element_by_id("su").click()
driver.close()
```

## 9.4.2 Opera

Opera 是 Blink 内核的浏览器，使用 `OperaChromiumDriver`。

GitHub 地址：<https://github.com/operasoftware/operachromiumdriver>

下载 `operadriver_win64.zip`，解压后得到 `operadriver.exe`，将其复制到 `C:\Python35` 目录下。

`baidu.py`

```
from selenium import webdriver

driver = webdriver.Opera()
driver.get("http://www.baidu.com")

driver.find_element_by_id("kw").send_keys("opera")
driver.find_element_by_id("su").click()
driver.quit()
```

## 9.4.3 Safari

Safari 是 Apple 公司的浏览器，只能在 MAC 系统上使用。Windows 系统可以使用 Selenium Server 的 `Remote` 功能。

WindowsLinuxSelenium Server

```
> java -jar selenium-server-standalone-2.47.0.jar
```

baidu.py

```
from selenium.webdriver import Remote
```

```
dc = {'browserName': 'safari'}
```

```
driver = Remote(command_executor=' http://127.0.0.1:4444/wd/hub',  
                desired_capabilities=dc)
```

```
driver.get("http://www.baidu.com")
```

```
driver.find_element_by_id("kw").send_keys("safari")
```

```
driver.find_element_by_id("su").click()
```

```
driver.quit()
```

## 9.4.4 HtmlUnit

HtmlUnit<http://htmlunit.sourceforge.net/>

HtmlUnitJavaHtmlUnit  
Java  
Selenium ServerHtmlUnit

WindowsLinuxSelenium Server

```
> java -jar selenium-server-standalone-2.47.0.jar
```





PhantomJS PhantomJS Windows MAC  
Linux

phantomjs.exe C:\Python35  
PhantomJS

baidu.py

```
from selenium import webdriver  
from time import sleep
```

```
driver = webdriver.PhantomJS()  
driver.get("http://www.baidu.com")
```

```
try:  
    driver.find_element_by_id("kw").send_keys("phantomjs")  
    driver.find_element_by_id("su").click()  
    sleep(1)  
    driver.get_screenshot_as_file("D:\\baidu_ok.jpg")  
except WebDriverException as msg:  
    print(msg)  
    driver.get_screenshot_as_file("D:\\baidu_error.jpg")  
finally:  
    driver.quit()
```

HtmlUnit PhantomJS  
D  
baidu\_ok.jpg 9.9

```

    selenium.grid.selenium.server
    selenium.server.remote
    selenium.remote

```



# 第10章 Python 自动化测试

---

在本章第9章 Selenium Grid 自动化测试中，我们介绍了 Selenium Grid 自动化测试的基本概念。在本章中，我们将介绍如何使用 Python 编写 Selenium 自动化测试脚本。

Python 是一种高级编程语言，具有简单易学、功能强大、社区活跃等特点。

在本章中，我们将介绍如何使用 Python 编写 Selenium 自动化测试脚本。

本章内容如下：

1. Selenium 自动化测试的基本概念  
2. Selenium 自动化测试的框架  
3. Selenium 自动化测试的脚本编写

本章内容如下：

1. Selenium 自动化测试的基本概念  
2. Selenium 自动化测试的框架  
3. Selenium 自动化测试的脚本编写

## 10.1 Selenium 自动化测试

在本章中，我们将介绍如何使用 Selenium 编写自动化测试脚本。我们将使用 Python 语言编写 Selenium 自动化测试脚本，并使用 Selenium 框架进行自动化测试。

## onethread.py

```
from time import sleep, ctime
```

```
# 音乐
```

```
def music():
```

```
    print('I was listening to music! %s' % ctime())
```

```
    sleep(2)
```

```
# 电影
```

```
def movie():
```

```
    print('I was at the movies! %s' % ctime())
```

```
    sleep(5)
```

```
if __name__ == '__main__':
```

```
    music()
```

```
    movie()
```

```
    print('all end:', ctime())
```

```
    music()movie()25sleep()
```

## Python Shell

```
===== RESTART: D:/thread_test/onethread.py
=====
```

I was listening to music! Sat Feb 14 20:11:04 2015

I was at the movies! Sat Feb 14 20:11:06 2015

all end: Sat Feb 14 20:11:11 2015

```

    1104music1106movie
1111movie7
```

```


```

```

musicmovie

```

onethread2.py

```
from time import sleep, ctime
```

```
#
```

```
def music(func, loop):
```

```
    for i in range(loop):
```

```
        print('I was listening to %s! %s' % (func, ctime()))
```

```
        sleep(2)
```

```
#
```

```
def movie(func, loop):
```



Python의 thread와 threading 모듈은 thread 모듈을  
사용하는 것보다 threading 모듈을 사용하는 것이 Lock 모듈을  
Condition 모듈을 사용하는 것보다 Python의  
성능이 더 좋다

## 10.2.1 threading

thread 모듈은 thread 모듈을 사용하는 것보다 threading 모듈을 사용하는 것이  
threading 모듈을 사용하는 것보다 threading 모듈을 사용하는 것이

threads.py

```
from time import sleep, ctime
import threading
```

```
# 음악
```

```
def music(func, loop):
    for i in range(loop):
        print("I was listening to %s! %s" % (func, ctime()))
        sleep(2)
```

```
# 영화
```

```
def movie(func, loop):
    for i in range(loop):
```



```
print("I was at the %s! %s" % (func, ctime()))
sleep(5)
```

```
# 初始化
```

```
threads = []
```

```
# 创建t1,初始化函数
```

```
t1 = threading.Thread(target=music, args=('音乐',2))
```

```
threads.append(t1)
```

```
# 创建t2,初始化函数
```

```
t2 = threading.Thread(target=movie, args=('电影', 2))
```

```
threads.append(t2)
```

```
if __name__ == '__main__':
```

```
    # 启动
```

```
    for t in threads:
```

```
        t.start()
```

```
    # 等待
```

```
    for t in threads:
```

```
        t.join()
```

```
    print('all end: %s' % ctime())
```

```
import threading
```

```
threads = []
```

```
threading.Thread()  # Create a threading.Thread() object
```

```
for threads in threads:
    threads.start()
threads.join()
print("all end")
```

### **class threading.Thread()**

```
class threading.Thread(group=None, target=None,
name=None, args=(), kwargs={})
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is the argument tuple for the target invocation.

Defaults to ().

`kwargs` is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If the subclass overrides the constructor, it must make sure to invoke the base class constructor (`Thread.__init__()`) before doing anything else to the thread.

□□□□

## Python Shell

```
===== RESTART:      D:/thread_test/threads.py
=====
```

```
I was listening to □□□□! Mon Nov  2 21:11:24 2015I was at the □
□□! Mon Nov
2 21:11:24 2015
```

```
I was listening to □□□□! Mon Nov  2 21:11:26 2015
I was at the □□□! Mon Nov  2 21:11:29 2015
all end: Mon Nov  2 21:11:34 2015
```

```
□□□□□□□□□□□□□□□□music□movie□□□□□□11□24□□□□□□
□□□□□11□34□□□□□10□□movie□□□□□□□□□10□□music□□□□□□□
4□□□□□□□□□□□□□□□□□□□□□□□□
```

## 10.2.2 线程池

线程池是一种多线程处理形式，处理过程中，线程数目可任意增减。它的实现，需要一个线程池的接口，一个线程池的实现类。接口如下：

player.py

```
from time import sleep, ctime
import threading

# 接口
def super_player(file_, time):
    for i in range(2):
        print('Start playing %s! %s' %(file_, ctime()))
        sleep(time)

# 实现类
lists = {'00000.mp3':3, '0000.mp4':5, '0000.mp3':4}

threads = []
files = range(len(lists))

# 实现
for file_, time in lists.items():
    t = threading.Thread(target=super_player, args=(file_,
time))
    threads.append(t)
```

```

if __name__ == '__main__':
    # 开始
    for t in files:
        threads[t].start()
    for t in files:
        threads[t].join()

    print('end: %s' % ctime())

    开始播放音乐文件
    开始播放音乐文件

    开始播放音乐文件
    for 开始播放音乐文件
    super_player() 开始播放音乐文件
    threads

    threads

```

## Python Shell

```

===== RESTART: D:/thread_test/player.py
=====

Start playing 开始播放音乐文件! Mon Nov 2 21:15:51 2015
Start playing 开始播放音乐文件! Mon
Nov 2 21:15:51 2015
Start playing 开始播放音乐文件! Mon Nov 2
21:15:51 2015

Start playing 开始播放音乐文件! Mon Nov 2 21:15:54 2015

```

```
Start playing 0000.mp3! Mon Nov  2 21:15:55 2015
Start playing 0000.mp4! Mon Nov  2 21:15:56 2015
end: Mon Nov  2 21:16:01 2015
```

## 10.2.3 多线程

用Python实现多线程

mythread.py

```
import threading
from time import sleep, ctime

# 多线程
class MyThread(threading.Thread):

    def __init__(self, func, args, name=''):
        threading.Thread.__init__(self)
        self.func = func
        self.args = args
        self.name = name

    def run(self):
        self.func(*self.args)
```

```

def super_play(file_, time):
    for i in range(2):
        print('Start playing %s! %s' % (file_, ctime()))
        sleep(time)

lists = {'0000.mp3':3, '0000.mp4':5, '0000.mp3':4}

threads = []
files = range(len(lists))

for file_, time in lists.items():
    t = MyThread(super_play, (file_, time),
super_play.__name__)
    threads.append(t)

if __name__ == '__main__':
    # 0000
    for i in files:
        threads[i].start()
    for i in files:
        threads[i].join()
    print('end:%s' % ctime())

```

```
MyThread(threading.Thread)
```

```
class MyThread(threading.Thread):
```

```
    def __init__(self, func, args, name):
```

Python 2의 `apply(func [, args [, kwargs ]])`는 함수를 호출하는 데 사용됩니다. `apply()`는 `args`와 `kwargs`를 사용하여 함수를 호출합니다.

Python 3에서는 `apply()`가 없습니다.

```
apply(self.func, self.args)
```

예를 들어

```
self.func(*self.args)
```

이 코드는 `MyThread` 클래스의 `run` 메서드를 호출합니다.

## 10.3 스레딩

### 10.3.1 multiprocessing

`multiprocessing` 모듈은 `threading` 모듈과 유사하지만, `multiprocessing`는 `Global Interpreter Lock, GIL`을 사용하지 않습니다. `GIL`은 CPU를 한 번에 하나씩만 사용할 수 있게 합니다. `multiprocessing`는 `UNIX`와 `Windows` 모두에서 작동합니다.

`threading` 모듈의 `Thread` 클래스와 `multiprocessing` 모듈의 `Process` 클래스를 비교해 보면



## process.py

```
from time import sleep, ctime
import multiprocessing

def super_player(file_, time):
    for i in range(2):
        print('Start playing %s! %s' %(file_, ctime()))
        sleep(time)

lists = {'0000.mp3':3, '000.mp4':5, '000.mp3':4}

threads = []
files = range(len(lists))

# 0000
for file_, time in lists.items():
    t = multiprocessing.Process(target=super_player, args=
(file_, time))
    threads.append(t)

if __name__ == '__main__':
    # 0000
    for t in files:
        threads[t].start()
    for t in files:
```





multiprocessing.Queue

multiprocessing.Pipe Queue IPC Pipe Queue

① Pipe half-duplex duplex multiprocessing.Pipe duplex=False pipe pipe

pipe.py

```
import multiprocessing
```

```
def proc1(pipe):  
    pipe.send('hello')  
    print('proc1 rec:', pipe.recv())
```

```
def proc2(pipe):  
    print('proc2 rec:', pipe.recv())  
    pipe.send('hello, too')
```

```
if __name__ == '__main__':  
    multiprocessing.freeze_support()  
    pipe = multiprocessing.Pipe()
```

```
p1 = multiprocessing.Process(target=proc1, args=(pipe[0],))
p2 = multiprocessing.Process(target=proc2, args=(pipe[1],))
```

```
p1.start()
p2.start()
p1.join()
p2.join()
```

```
pipe = multiprocessing.Pipe()
proc1 = multiprocessing.Process(target=proc1, args=(pipe[0],))
proc2 = multiprocessing.Process(target=proc2, args=(pipe[1],))
proc1.start()
proc2.start()
proc1.join()
proc2.join()
pipe.close()
```

## Python Shell

```
===== RESTART: D:/thread_test/pipe.py
=====
proc2 rec: hello
proc1 rec: hello, too
```

```
② Queue和Pipe
Queue和Pipe都是线程安全的，Queue是线程安全的，Pipe是线程安全的。
Queue和Pipe都是线程安全的，Queue是线程安全的，Pipe是线程安全的。
Queue和Pipe都是线程安全的，Queue是线程安全的，Pipe是线程安全的。
```

## queue.py

```
import multiprocessing
import os, time
```

```

def inputQ(queue):
    info = str(os.getpid()) + '(put):' + str(time.time())
    queue.put(info)

def outputQ(queue, lock):
    info = queue.get()
    lock.acquire()
    print((str(os.getpid()) + '(get):' + info))
    lock.release()

if __name__ == '__main__':

    record1 = []
    record2 = []
    lock = multiprocessing.Lock()    # 互斥锁
    queue = multiprocessing.Queue(3)

    for i in range(10):
        process = multiprocessing.Process(target=inputQ, args=
(queue,))
        process.start()
        record1.append(process)

    for i in range(10):
        process = multiprocessing.Process(target=outputQ, args=
(queue, lock))

```

```

        process.start()
        record2.append(process)

for p in record1:
    p.join()

queue.close() # 队列关闭

for p in record2:
    p.join()

print

```

## Python Shell

```

===== RESTART: D:/thread_test/queue.py
=====

784(get):4988(put):1445747594.7366996
5460(get):2344(put):1445747594.7679493
4752(get):2648(put):1445747594.83045
9172(get):10964(put):1445747594.83045
9660(get):10888(put):1445747594.8460755
6300(get):5292(put):1445747594.9710765
7232(get):6960(put):1445747595.002327
11052(get):7992(put):1445747595.002327
5384(get):260(put):1445747595.0179522
8448(get):6204(put):1445747595.0179522

```

IPC

## 10.4

[illegible]

### 10.4.1 □□□□□□□□

[illegible]

# baidu\_thread.py

```
from threading import Thread
from selenium import webdriver
from time import ctime, sleep

# 测试百度

def test_baidu(browser, search):
    print('start:%s' % ctime())
    print('browser:%s ,' % browser)
    if browser == "ie":
        driver = webdriver.Ie()
    elif browser == "chrome":
```



```

        driver = webdriver.Chrome()
    elif browser == "ff":
        driver = webdriver.Firefox()
    else:
        print("browser\t\t\t\t\tie\tff\tchrome")

    driver.get('http://www.baidu.com')
    driver.find_element_by_id("kw").send_keys(search)
    driver.find_element_by_id("su").click()
    sleep(2)
    driver.quit()

if __name__ == '__main__':
    # 创建线程池
    lists = {'chrome': 'threading', 'ie': 'webdriver',
'ff': 'python'}

    threads = []
    files = range(len(lists))

    # 开始
    for browser, search in lists.items():
        t = Thread(target=test_baidu, args=(browser, search))
        threads.append(t)

    # 结束
    for t in files:

```



Grid

## Selenium Server

hub node 4444 5555 IP  
172.16.10.66

```
C:\selenium>java -jarselenium-server-standalone-2.47.0.jar -  
role hub
```

```
C:\selenium>java -jarselenium-server-standalone-2.47.0.jar -  
role node -port 5555
```

node 6666 IP 172.16.10.34

```
fnngj@fnngj-VirtualBox:~/selenium$ java -jarselenium-server-  
standalone-2.47.0.jar -role node -port 6666 -hub  
http://172.16.10.66:4444/grid/register
```

grid\_thread.py

```
from threading import Thread  
from selenium import webdriver  
from time import sleep, ctime
```

```

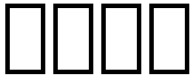
# 测试
def test_baidu(host,browser):
    print('start:%s' % ctime())
    print(host, browser)
    dc = {'browserName': browser}
    driver = webdriver.Remote(command_executor=host,
                              desired_capabilities=dc)
    driver.get('http://www.baidu.com')
    driver.find_element_by_id("kw").send_keys(browser)
    driver.find_element_by_id("su").click()
    driver.close()

if __name__ == '__main__':
    # 测试用例
    lists = {'http://127.0.0.1:4444/wd/hub': 'chrome',
             'http://127.0.0.1:5555/wd/hub': 'internet
explorer',
             'http://172.16.10.34:6666/wd/hub': 'firefox', # 测试用例
    }
    threads = []
    files = range(len(lists))

    # 测试
    for host, browser in lists.items():
        t = Thread(target=test_baidu, args=(host, browser))

```





Python

unittest Java  
TestNG



# 11 数据库系统

---

数据库系统是由数据库、数据库管理系统、数据库管理员、数据库用户、数据库应用程序等组成的。数据库是存储在计算机中的有组织的数据集合。数据库管理系统是管理数据库的软件。数据库管理员是负责数据库系统的管理和维护的人员。数据库用户是使用数据库系统的人员。数据库应用程序是运行在数据库系统上的应用程序。

数据库系统的主要功能包括：数据定义、数据操作、数据控制、数据查询、数据维护等。数据库系统的主要特点包括：数据共享、数据独立性、数据安全性、数据完整性、数据一致性等。

## 11.1 数据库系统组成

数据库系统是由数据库、数据库管理系统、数据库管理员、数据库用户、数据库应用程序等组成的。数据库是存储在计算机中的有组织的数据集合。数据库管理系统是管理数据库的软件。数据库管理员是负责数据库系统的管理和维护的人员。数据库用户是使用数据库系统的人员。数据库应用程序是运行在数据库系统上的应用程序。

### 11.1.1 数据库系统组成

数据库系统是由数据库、数据库管理系统、数据库管理员、数据库用户、数据库应用程序等组成的。数据库是存储在计算机中的有组织的数据集合。数据库管理系统是管理数据库的软件。数据库管理员是负责数据库系统的管理和维护的人员。数据库用户是使用数据库系统的人员。数据库应用程序是运行在数据库系统上的应用程序。数据库系统的主要功能包括：数据定义、数据操作、数据控制、数据查询、数据维护等。数据库系统的主要特点包括：数据共享、数据独立性、数据安全性、数据完整性、数据一致性等。

数据库系统的主要功能包括：数据定义、数据操作、数据控制、数据查询、数据维护等。



# 1. 目的と意義

- 本報告書の目的は、プロジェクトの進捗状況を把握し、関係者への情報提供を行うことである。
- 本報告書の意義は、プロジェクトの透明性を高め、関係者の理解と協力を促進することである。
- 本報告書の対象者は、プロジェクトの関係者、特にプロジェクトマネージャー、関係者、関係者である。
- 本報告書の範囲は、プロジェクトの進捗状況、関係者の役割、関係者の役割である。

# 2. 背景と現状

- 本プロジェクトは、2023年1月1日より開始された。
- 本プロジェクトの目的は、プロジェクトの進捗状況を把握し、関係者への情報提供を行うことである。
- 本プロジェクトの意義は、プロジェクトの透明性を高め、関係者の理解と協力を促進することである。
- 本プロジェクトの対象者は、プロジェクトの関係者、特にプロジェクトマネージャー、関係者、関係者である。

本プロジェクトは、2023年1月1日より開始された。本プロジェクトの目的は、プロジェクトの進捗状況を把握し、関係者への情報提供を行うことである。本プロジェクトの意義は、プロジェクトの透明性を高め、関係者の理解と協力を促進することである。本プロジェクトの対象者は、プロジェクトの関係者、特にプロジェクトマネージャー、関係者、関係者である。

本プロジェクトは、2023年1月1日より開始された。本プロジェクトの目的は、プロジェクトの進捗状況を把握し、関係者への情報提供を行うことである。本プロジェクトの意義は、プロジェクトの透明性を高め、関係者の理解と協力を促進することである。本プロジェクトの対象者は、プロジェクトの関係者、特にプロジェクトマネージャー、関係者、関係者である。

# 3. 結論と今後の展望

1. 本プロジェクトの進捗状況を把握し、関係者への情報提供を行うことである。

2. 本プロジェクトの意義は、プロジェクトの透明性を高め、関係者の理解と協力を促進することである。

関係者

3. 在代码中，我们使用 `std::cout` 来输出结果。这里我们使用了 `<string>` 头文件，所以我们可以使用 `std::string` 类型。

4. 在代码中，我们使用 `std::endl` 来换行。这里我们使用了 `<string>` 头文件，所以我们可以使用 `std::string` 类型。

5. 在代码中，我们使用 `std::endl` 来换行。这里我们使用了 `<string>` 头文件，所以我们可以使用 `std::string` 类型。

6. 在代码中，我们使用 `std::endl` 来换行。这里我们使用了 `<string>` 头文件，所以我们可以使用 `std::string` 类型。

7. 在代码中，我们使用 `std::endl` 来换行。这里我们使用了 `<string>` 头文件，所以我们可以使用 `std::string` 类型。

8. 在代码中，我们使用 `std::endl` 来换行。这里我们使用了 `<string>` 头文件，所以我们可以使用 `std::string` 类型。

## 11.1.2 字符串

### 1. 字符串

在 C++ 中，字符串通常使用 `std::string` 类型。这里我们使用了 `<string>` 头文件，所以我们可以使用 `std::string` 类型。

- 在代码中，我们使用 `std::string` 来声明字符串变量。这里我们使用了 `<string>` 头文件，所以我们可以使用 `std::string` 类型。
- 在代码中，我们使用 `std::string` 来声明字符串变量。这里我们使用了 `<string>` 头文件，所以我们可以使用 `std::string` 类型。
- 在代码中，我们使用 `std::string` 来声明字符串变量。这里我们使用了 `<string>` 头文件，所以我们可以使用 `std::string` 类型。
- 在代码中，我们使用 `std::string` 来声明字符串变量。这里我们使用了 `<string>` 头文件，所以我们可以使用 `std::string` 类型。

[illegible]

**2** □ □ □ □ □

Web

**3**

[illegible][illegible]

**4**

HTML `<div id="name">`

Web doc3861

doc6148 “ ”  
Web

## 5 Ajax

Ajax  
RSS Ajax  
Ajax

### 11.1.3

- 
- 
- 
- 
- 

## 11.2 BBS

[illegible]

## 11.2.1 ☐☐☐☐

**1**

[illegible][illegible][illegible]

## 2000 IDE

Python  
IDE Python IDLE Python

Python IDE

Python IDLE Python IDE  
IDE Shell Python Python

UliPad Python IDE wxPython

`Sublime``IDE``Ctrl+d`

PyCharm Python IDE

Eclipse + pydev = Eclipse IDE for Java  
 IDE + pydev = Python IDE Eclipse

Vim Emacs

IDE

## 11.2.2 ☐☐☐☐☐☐

# 11.1

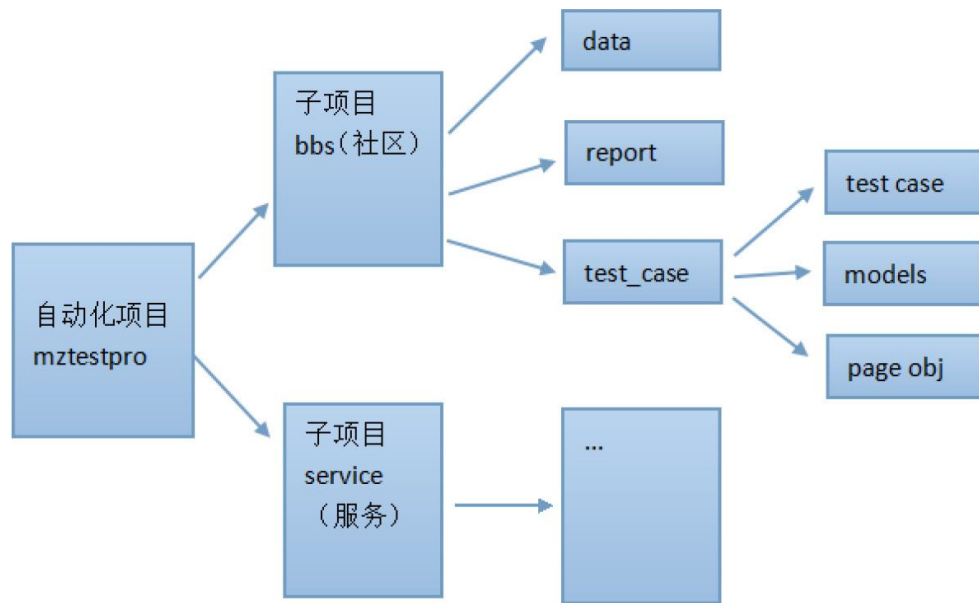


图11.1 项目结构图

项目结构图

```

mztestpro/
├─ bbs/
│   ├── data/
│   ├── report/
│   │   └─ image/
│   └─ test_case/
│       ├── models/
│       │   ├── driver.py
│       │   ├── function.py
│       │   └─ myunit.py
│       ├── page_obj/
│       │   └─ *Page.py
│       └─ *_sta.py
└─ driver/
  
```

- └─ package/
- └─ run\_bbs\_test.py
- └─ startup.bat
- └─ 測試測試測試.docx

## 1 mztestpro

bbs 測試 BBS 測試測試測試測試測試測試

driver 測試測試測試 selenium-server-standalone-2.47.0.jar chromedriver.exe IEDriverServer.exe 測試測試測試測試測試測試測試測試測試測試測試測試測試 path 測試

package 測試測試測試測試測試測試 HTMLTestRunner.py 測試測試測試測試測試測試測試測試測試測試測試 Python Lib 測試

run\_bbs\_test.py 測試測試測試測試測試 BBS 測試測試

startup.bat 測試 Selenium Server 測試 driver 測試 selenium-server-standalone-2.47.0.jar

測試測試測試.docx 測試測試測試測試測試測試

## 2 bbs

data 測試測試測試測試測試

report 測試 HTML 測試測試測試 image 測試測試測試測試測試



test\_case

### 3 test\_case

models

page\_obj Page Object  
“\*Page.py”

\*\_sta.py “\*\_sta.py”

## 11.2.3

```
...\mztestpro\bbs\test_case\models\driver.py
```

driver.py

```
from selenium.webdriver import Remote  
from selenium import webdriver
```

```
#
```

```
def browser():
```

```
    # driver = webdriver.Chrome()
```

```

        host = '127.0.0.1:4444'      # 本地地址 127.0.0.1:4444
        dc = {'browserName': 'chrome'} # 浏览器名称
        ['chrome','firefox'],
        driver = Remote(command_executor='http://' + host +
                          '/wd/hub',
                          desired_capabilities=dc)
    return driver

```

```

if __name__ == '__main__':

```

```

    dr = browser()
    dr.get("http://www.baidu.com")
    dr.quit()

```

```

    # 运行浏览器
    browser()
    # 运行时间
    10

```

```

    # 运行时间

```

```

... \mztestpro\bbs\test_case\models\myunit.py

```

myunit.py

```

from selenium import webdriver
from .driver import browser
import unittest
import os

```

```
class MyTest(unittest.TestCase):
```

```
    def setUp(self):
```

```
        self.driver = browser()
```

```
        self.driver.implicitly_wait(10)
```

```
        self.driver.maximize_window()
```

```
    def tearDown(self):
```

```
        self.driver.quit()
```

```
    def __init__(self, *args, **kwargs):
        super(MyTest, self).__init__(*args, **kwargs)
        self.setUp()
        self.tearDown()
        self.driver = browser()
        self.driver.implicitly_wait(10)
        self.driver.maximize_window()
```

```
    def __del__(self):
```

```
... \mztestpro\bbs\test_case\models\function.py
```

function.py

```
from selenium import webdriver
```

```
import os
```

```
# 测试
```

```
def insert_img(driver, file_name):
```

```
    base_dir = os.path.dirname(os.path.dirname(__file__))
```

```

base_dir = str(base_dir)
base_dir = base_dir.replace('\\', '/')
base = base_dir.split('/test_case')[0]
file_path = base + "/report/image/" + file_name
driver.get_screenshot_as_file(file_path)

```

```

if __name__ == '__main__':
    driver = webdriver.Chrome()
    driver.get("https://www.baidu.com")
    insert_img(driver, 'baidu.jpg')
    driver.quit()

```

```

def insert_img(driver, file_name):
    file_path = os.path.join(
        os.getcwd(), 'report', 'image', file_name)
    driver.get_screenshot_as_file(file_path)

```

## 11.2.4 Page Object

Page Object 模式是 8.3 章节中提到的模式之一，它主要用于提高测试代码的可维护性和复用性。

Page Object 模式

```

... \mztestpro\bbs\test_case\page_obj\base.py

```

base.py

```

class Page(object):
    '''
    页面抽象基类
    '''

    bbs_url = 'http://bbs.meizu.cn'

    def __init__(self, selenium_driver, base_url=bbs_url,
parent=None):
        self.base_url = base_url
        self.driver = selenium_driver
        self.timeout = 30
        self.parent = parent

    def _open(self,url):
        url = self.base_url + url
        self.driver.get(url)
        assert self.on_page(),'Did not land on %s' % url

    def find_element(self, *loc):
        return self.driver.find_element(*loc)

    def find_elements(self, *loc):
        return self.driver.find_elements(*loc)

    def open(self):
        self._open(self.url)

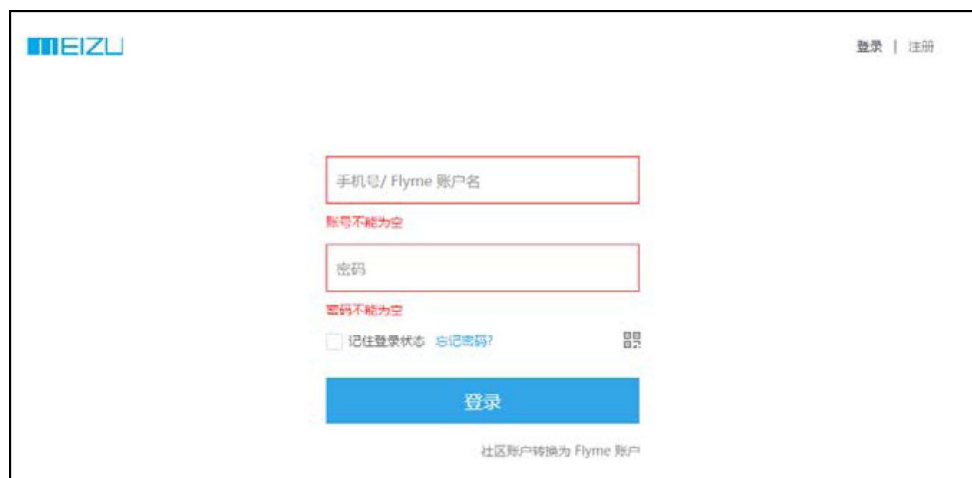
```

```
def on_page(self):
    return self.driver.current_url == (self.base_url +
self.url)
```

```
def script(self,src):
    return self.driver.execute_script(src)
```

在\_\_init\_\_()中设置初始URL，并调用open()、BBS.find\_element()、find\_elements()、script()、JavaScript等函数，WebDriver

11.2



11.2

BBS

...\mztestpro\bbs\test\_case\page\_obj\loginPage.py

## loginPage.py

```
from selenium.webdriver.common.action_chains import
ActionChains
from selenium.webdriver.common.by import By
from .base import Page
from time import sleep

class login(Page):
    '''
    登陆页面
    '''

    url = '/'

    # Action
    bbs_login_user_loc = (By.XPATH,
        "//div[@id='mzCust']/div/img")
    bbs_login_button_loc = (By.ID, "mzLogin")

    def bbs_login(self):
        self.find_element(*self.bbs_login_user_loc).click()
        sleep(1)
        self.find_element(*self.bbs_login_button_loc).click()

    login_username_loc = (By.ID, "account")
```

```

login_password_loc = (By.ID, "password")
login_button_loc = (By.ID, "login")

# 用户名
def login_username(self, username):

self.find_element(*self.login_username_loc).send_keys(username)

# 密码
def login_password(self, password):

self.find_element(*self.login_password_loc).send_keys(password)

# 登录
def login_button(self):
    self.find_element(*self.login_button_loc).click()

# 用户登录
def user_login(self, username="username", password="1111"):
    """ 用户登录 """
    self.open()
    self.bbs_login()
    self.login_username(username)
    self.login_password(password)
    self.login_button()
    sleep(1)

```



```

user_error_hint_loc = (By.XPATH, "//span[@for='account']" )
pwd_error_hint_loc = (By.XPATH, "//span[@for='password']")
user_login_success_loc = (By.ID, "mzCustName")

# 用户登录失败
def user_error_hint(self):
    return
self.find_element(*self.user_error_hint_loc).text

# 密码错误提示
def pwd_error_hint(self):
    return
self.find_element(*self.pwd_error_hint_loc).text

# 用户登录成功
def user_login_success(self):
    return
self.find_element(*self.user_login_success_loc).text

# 测试用例
def test_user_login(self):
    # 打开浏览器并访问登录页面
    self.driver.get("http://localhost:8080/mzCustName")
    # 输入用户名和密码
    self.driver.find_element(By.ID, "username").send_keys("admin")
    self.driver.find_element(By.ID, "password").send_keys("123456")
    # 点击登录按钮
    self.driver.find_element(By.ID, "login").click()
    # 断言登录成功
    self.assertEqual(self.driver.find_element(By.ID, "mzCustName").text, "admin")

```

## 11.2.5 测试用例

测试用例是测试人员在测试之前，根据需求规格说明书，设计出的测试通过的条件和测试失败的条件的集合。

测试BBS系统

```
...\mztestpro\bbs\test_case\login_sta.py
```

测试用例是测试人员在测试之前，根据需求规格说明书，设计出的测试通过的条件和测试失败的条件的集合。

测试用例login\_sta.py是测试人员在测试之前，根据需求规格说明书，设计出的测试通过的条件和测试失败的条件的集合。

测试用例“\*Page.py”是测试人员在测试之前，根据需求规格说明书，设计出的测试通过的条件和测试失败的条件的集合。

login\_sta.py

```
from time import sleep
import unittest, random, sys
sys.path.append("./models")
sys.path.append("./page_obj")
from models import myunit, function
from page_obj.loginPage import login
```

```
class loginTest(myunit.MyTest):
    '''测试用例'''

    # 测试用例
    def user_login_verify(self, username="", password=""):
```

```
login(self.driver).user_login(username, password)
```

```
def test_login1(self):
```

```
    '''测试用户名和密码为空'''
```

```
    self.user_login_verify()
```

```
    po = login(self.driver)
```

```
    self.assertEqual(po.user_error_hint(), "用户名或密码为空")
```

```
    self.assertEqual(po.pawd_error_hint(), "用户名或密码为空")
```

```
    function.insert_img(self.driver, "user_pawd_empty.jpg")
```

```
def test_login2(self):
```

```
    '''测试用户名不为空,密码为空'''
```

```
    self.user_login_verify(username="pytest")
```

```
    po = login(self.driver)
```

```
    self.assertEqual(po.pawd_error_hint(), "用户名或密码为空")
```

```
    function.insert_img(self.driver, "pawd_empty.jpg")
```

```
def test_login3(self):
```

```
    '''测试用户名不为空,密码不为空'''
```

```
    self.user_login_verify(password="abc123456")
```

```
    po = login(self.driver)
```

```
    self.assertEqual(po.user_error_hint(), "用户名或密码为空")
```

```
    function.insert_img(self.driver, "user_empty.jpg")
```

```
def test_login4(self):
```

```
    '''测试用户名和密码为随机字符'''
```

```
    character = random.choice('zyxwvutsrqponmlkjihgfedcba')
```

```

        username = "zhangsan" + character
        self.user_login_verify(username=username,
password="123456")
        po = login(self.driver)
        self.assertEqual(po.pawd_error_hint(), "密码错误")
        function.insert_img(self.driver, "user_pawd_error.jpg")

def test_login5(self):
    '''密码错误'''

self.user_login_verify(username="zhangsan",password="123456")
    sleep(2)
    po = login(self.driver)
    self.assertEqual(po.user_login_success(), '否')
    function.insert_img(self.driver, "user_pawd_ture.jpg")

if __name__ == "__main__":
    unittest.main()

```

```

    def loginTest():
        myunit.MyTest()
    def setUp():
        tearDown()

```

```




    def user_login_verify():
        loginPage.py
    def user_login():
        user_login()

```

```

    def setUp():

```

- 
- 
- 

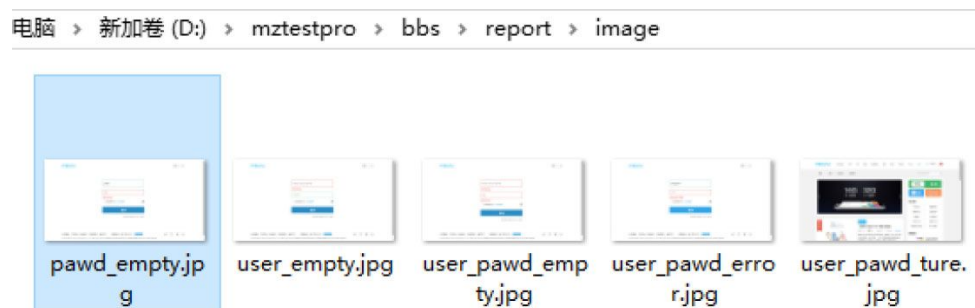
“zhangsan” 张 三 a~z

□ □

```

function.py
insert_img
.../report/image/11.3

```



□11.3 □□□□□□

### 11.2.6 ☐ ☐ ☐ ☐ ☐ ☐

`Selenium Grid`

```

    ...\\mztestpro\\startup.bat
    ...\\mztestpro\\driver\\
Selenium Server

```

startup.bat

```
java -jar ./driver/selenium-server-standalone-2.47.0.jar -role  
hub
```

```
strtp.batSelenium Serverhub  
Selenium Servernode9.3
```

...\mztestpro\run\_bbs\_test.py

run\_bbs\_test.py

```
from HTMLTestRunner import HTMLTestRunner  
from email.mime.text import MIMEText  
from email.header import Header  
import smtplib  
import unittest  
import time  
import os
```

```
# =====
```

```
def send_mail(file_new):
```

```
    f = open(file_new, 'rb')
```

```
    mail_body = f.read()
```

```
    f.close()
```

```
    msg = MIMEText(mail_body, 'html', 'utf-8')
```

```

msg['Subject'] = Header("测试邮件", 'utf-8')

smtp = smtplib.SMTP()
smtp.connect("smtp.126.com")
smtp.login("username@126.com", "123456")
smtp.sendmail("username@126.com", "receive@126.com",
msg.as_string())
smtp.quit()
print('email has send out !')

```

```

# =====测试邮件=====
def new_report(testreport):
    lists = os.listdir(testreport)
    lists.sort(key=lambda fn: os.path.getmtime(testreport +
"\\" + fn))
    file_new = os.path.join(testreport, lists[-1])
    print(file_new)
    return file_new

```

```

if __name__ == '__main__':
    now = time.strftime("%Y-%m-%d %H_%M_%S")
    filename = './bbs/report/' + now + 'result.html'
    fp = open(filename, 'wb')
    runner = HTMLTestRunner(stream=fp,
                            title='测试邮件',

```

```
description='Windows 7 浏览器  
chrome')
```

```
discover =  
unittest.defaultTestLoader.discover('./bbs/test_case',
```

```
pattern='*_sta.py')
```

```
runner.run(discover)
```

```
fp.close() # 关闭文件
```

```
file_path = new_report('./bbs/report/') # 生成报告
```

```
send_mail(file_path) # 发送邮件
```

```
HTMLTestRunner.HTMLTestRunner(HTMLTestRunner)  
HTMLTestRunner.run()
```

```
...models\driver.py  
run_bbs_test.py
```

测试

```
HTMLTestRunner.HTMLTestRunner(HTMLTestRunner)  
HTMLTestRunner.run()  
HTMLTestRunner.run()
```

BBS 11.4



魅族社区自动化测试报告

Start Time: 2015-07-28 10:33:31  
Duration: 0:45:19.369000  
Status: Pass 201

环境: windows 7 浏览器: chrome

Show [Summary](#) [Failed](#) [All](#)

| Test Group/Test case                                  | Count | Pass | Fail | Error | View                   |
|---|-------|------|------|-------|------------------------|
| atest_login_sta.loginTest: 社区登录测试                     | 5     | 5    | 0    | 0     | <a href="#">Detail</a> |
| bbs_home_page_sta.homePageFirmwareGet: 社区首页--固件下载     | 2     | 2    | 0    | 0     | <a href="#">Detail</a> |
| bbs_home_page_sta.homePageLink: 社区首页--板块链接测试          | 12    | 12   | 0    | 0     | <a href="#">Detail</a> |
| bbs_home_page_sta.homePageStatisticsTest: 社区首页--统计    | 3     | 3    | 0    | 0     | <a href="#">Detail</a> |
| bbs_home_page_sta.homePageTset: 社区首页--主要功能测试          | 5     | 5    | 0    | 0     | <a href="#">Detail</a> |
| bbs_home_page_sta.homePageWeiboTest: 社区首页--社区微博       | 1     | 1    | 0    | 0     | <a href="#">Detail</a> |
| bbs_sign_sta.bbsSignTset: 社区--签到功能测试                  | 3     | 3    | 0    | 0     | <a href="#">Detail</a> |
| bbs_sreach_sta.bbsSreachTset: 社区--搜索功能测试              | 5     | 5    | 0    | 0     | <a href="#">Detail</a> |
| my_dynamic_sta.dating: 个人中心--我的动态--大厅标签页测试            | 4     | 4    | 0    | 0     | <a href="#">Detail</a> |
| my_dynamic_sta.guangbo: 个人中心--我的动态--广播标签页测试           | 4     | 4    | 0    | 0     | <a href="#">Detail</a> |
| my_dynamic_sta.guangbo_shu: 个人中心--我的动态--用户广播数测试       | 2     | 2    | 0    | 0     | <a href="#">Detail</a> |
| my_dynamic_sta.guanzhu: 个人中心--我的动态--关注标签页测试           | 4     | 4    | 0    | 0     | <a href="#">Detail</a> |
| my_friend_sta.Friend: 个人中心--我的好友---好友                 | 3     | 3    | 0    | 0     | <a href="#">Detail</a> |
| my_friend_sta.Sreach: 个人中心--我的好友---搜索                 | 3     | 3    | 0    | 0     | <a href="#">Detail</a> |
| my_friend_sta.listenIn: 个人中心--我的好友---收听               | 1     | 1    | 0    | 0     | <a href="#">Detail</a> |
| my_friend_sta.listenOut: 个人中心--我的好友---听众              | 2     | 2    | 0    | 0     | <a href="#">Detail</a> |
| my_invitation_sta.Status: 个人中心--我的帖子--主题--状态测试        | 4     | 4    | 0    | 0     | <a href="#">Detail</a> |
| my_invitation_sta.otherTags: 个人中心--我的帖子--其它标签页        | 3     | 3    | 0    | 0     | <a href="#">Detail</a> |
| my_invitation_sta.selectPlate: 个人中心--我的帖子--主题--选择板块测试 | 8     | 8    | 0    | 0     | <a href="#">Detail</a> |
| my_message_sta.personMessage: 个人中心--我的消息---个人信息       | 5     | 5    | 0    | 0     | <a href="#">Detail</a> |
| my_message_sta.remindTest: 个人中心--我的消息---提醒            | 3     | 3    | 0    | 0     | <a href="#">Detail</a> |
| my_message_sta.systemMessage: 个人中心--我的消息---系统消息       | 2     | 2    | 0    | 0     | <a href="#">Detail</a> |
| my_setting_sta.activityInfo: 个人中心--我的设置---活动信息        | 5     | 5    | 0    | 0     | <a href="#">Detail</a> |
| my_setting_sta.baseInfo: 个人中心--我的设置---基本资料            | 12    | 12   | 0    | 0     | <a href="#">Detail</a> |
| my_setting_sta.integral: 个人中心--我的设置---积分              | 1     | 1    | 0    | 0     | <a href="#">Detail</a> |
| my_setting_sta.modifyAvatar: 个人中心--我的设置---修改头像        | 1     | 1    | 0    | 0     | <a href="#">Detail</a> |
| my_setting_sta.personalInfo: 个人中心--我的设置---个人信息        | 6     | 6    | 0    | 0     | <a href="#">Detail</a> |
| my_setting_sta.privacy: 个人中心--我的设置---隐私筛选             | 5     | 5    | 0    | 0     | <a href="#">Detail</a> |
| my_setting_sta.professionInfo: 个人中心--我的设置---职业信息      | 6     | 6    | 0    | 0     | <a href="#">Detail</a> |
| my_setting_sta.userGroup: 个人中心--我的设置---用户组            | 3     | 3    | 0    | 0     | <a href="#">Detail</a> |
| my_vest_sta.myvestTset: 个人中心--我的马甲                    | 6     | 6    | 0    | 0     | <a href="#">Detail</a> |

11.4 BBS



# 第12章 BDD与Lettuce

本章主要介绍BDD（Behavior Driven Development）测试方法，以及Lettuce测试框架。BDD是一种测试方法，它强调测试用例的编写应该与业务需求紧密相关。Lettuce是一个Python 3的BDD测试框架，它支持Python 2和Python 3，并且可以与Python 3的unittest库集成。

本章主要介绍Lettuce测试框架，以及Python 3的BDD测试框架。Lettuce是一个Python 3的BDD测试框架，它支持Python 2和Python 3，并且可以与Python 3的unittest库集成。

## 12.1 什么是BDD

BDD（Behavior Driven Development）是一种测试方法，它强调测试用例的编写应该与业务需求紧密相关。BDD测试方法通常包括以下几个步骤：

### 1. TDD（Test-Driven Development）

TDD（Test-Driven Development）是一种测试方法，它强调测试用例的编写应该与业务需求紧密相关。TDD测试方法通常包括以下几个步骤：

### 2. ATDD（Acceptance Test Driven Development）

ATDD（Acceptance Test Driven Development）是一种测试方法，它强调测试用例的编写应该与业务需求紧密相关。ATDD测试方法通常包括以下几个步骤：

### 3 BDD Behavior Driven Development

QA BDD Dan North 2003

BDD

Cucumber Ruby <https://cucumber.io>

Jdave Java <http://jdave.org>

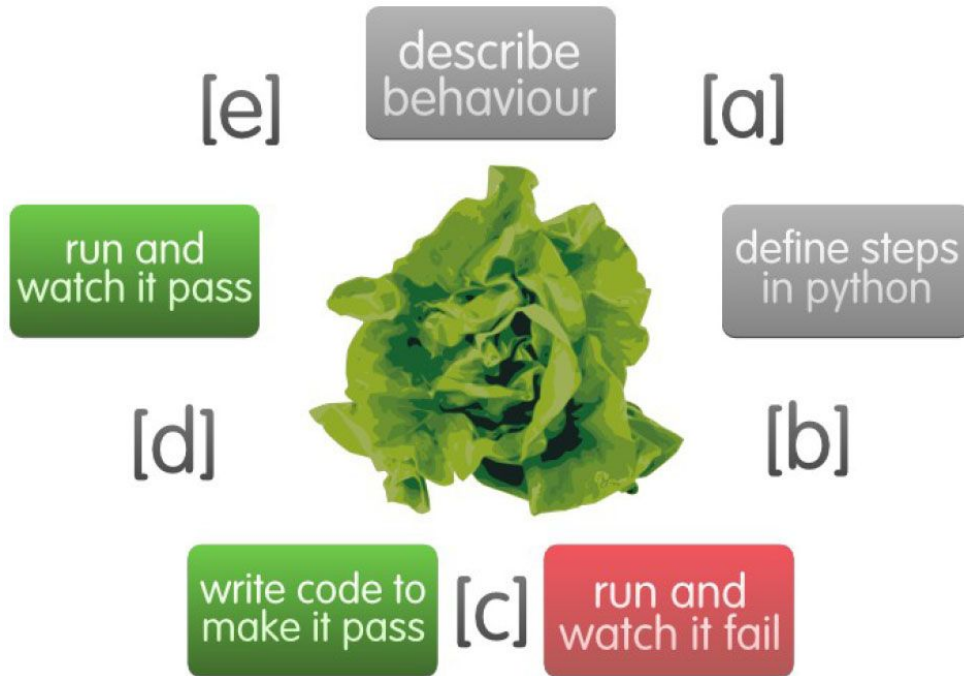
Behat PHP <http://docs.behat.org/en/v2.5>

Behave Python <http://pythonhosted.org/pytest>

Lettuce Python <http://Lettuce.it>

Ruby Cucumber BDD Python Cucumber Python BDD Lettuce

Lettuce 12.1



12.1 Lettuce workflow

- [a] Define steps
- [b] Run Python code
- [c] Write code to make it pass
- [d] Run code and watch it pass
- [e] Describe behaviour

## 12.2 Installing Lettuce

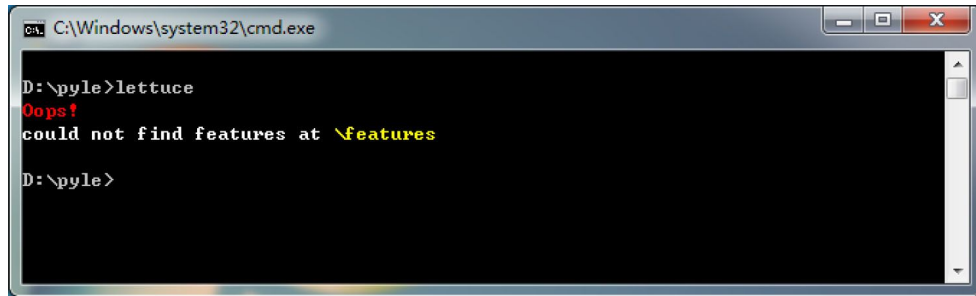
Lettuce website: <http://Lettuce.it/>

Install Lettuce using pip:

cmd.exe

```
> pip install lettuce
```

Lettuces in Windows are called "Lettuce" in 12.2



12.2 Lettuces

The "could not find features at \features" message means features are not installed. Lettuces are installed in 12.2.

## 12.3

Lettuces are installed

### 12.3.1

factorial is defined as 0! = 1

$$0! = 1$$

$$1! = 1$$

$$2! = 2 \times 1 = 2$$

$$3! = 3 \times 2 \times 1 = 6$$

.....

$$10! = 10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 3628800$$

.....

用Python实现阶乘

factorial.py

# 用for循环

```
def f1(n):  
    c = 1  
    for i in range(n):  
        i = i + 1  
        c = c * i  
    return c
```

# 用递归

```
def f2(n):  
    if n > 1:  
        return n*f2(n-1)  
    else:
```

```
return 1
```

```
if __name__ == '__main__':
```

```
    # 测试
```

```
    print(f1(10))
```

```
    print(f2(10))
```

```
    测试BDD测试
```

## 12.3.2 测试BDD测试

```
测试测试
```

```
projects/mymath/tests/
```

```
    └─ features/
```

```
        └─ zero.feature
```

```
        └─ steps.py
```

```
测试测试zero.feature 测试测试
```

```
zero.feature
```

**Feature:**

Compute factorial

In order to play with Lettuce

As beginners

We'll implement factorial



## Scenario:

Factorial of 0

### Given

I have the number 0

### When

I compute its factorial

### Then

I see the number 1

```
zero.feature
```

```
zero.feature
```

```
    
```

```
    
```

```
        lettuce
```

```
        
```

```
        
```

```
    
```

```
0    
```

0

1

Lettuce

- Feature
- Scenario
- Given
- And
- When
- Then

Lettuce 12.1

12.1 Lettuce unittest

| Lettuce | unittest   |
|---------|------------|
| Feature | test suite |
|         |            |

|          |           |
|----------|-----------|
| Scenario | test case |
| Given    | setup     |
| When     | test run  |
| Then     | assert    |

zero.feature steps.py

steps.py

```
from lettuce import *
```

```
@step('I have the number (\d+)')
```

```
def have_the_number(step, number):
```

```
    world.number = int(number)
```

```
@step('I compute its factorial')
```

```
def compute_its_factorial(step):
```

```
    world.number = factorial(world.number)
```

```
@step('I see the number (\d+)')
```

```
def check_number(step, expected):
```

```
    expected = int(expected)
```

```
    assert world.number == expected, "Got %d" % world.number
```



② `@step(I have the number (\d+))` `0` `int` `world.number`

```
@step('I compute its factorial')
def compute_its_factorial(step):
    world.number = factorial(world.number)
```

③ `have_the_number()` `world.number` `0` `factorial()` `factorial()` `world.number`

`I compute its factorial` `zero.feature` `"When I compute its factorial"`

```
def factorial(number):
    number = int(number)
    if (number == 0) or (number == 1):
        return 1
    else:
        return number
```

④ `0` `1` `1` `have_the_number()`

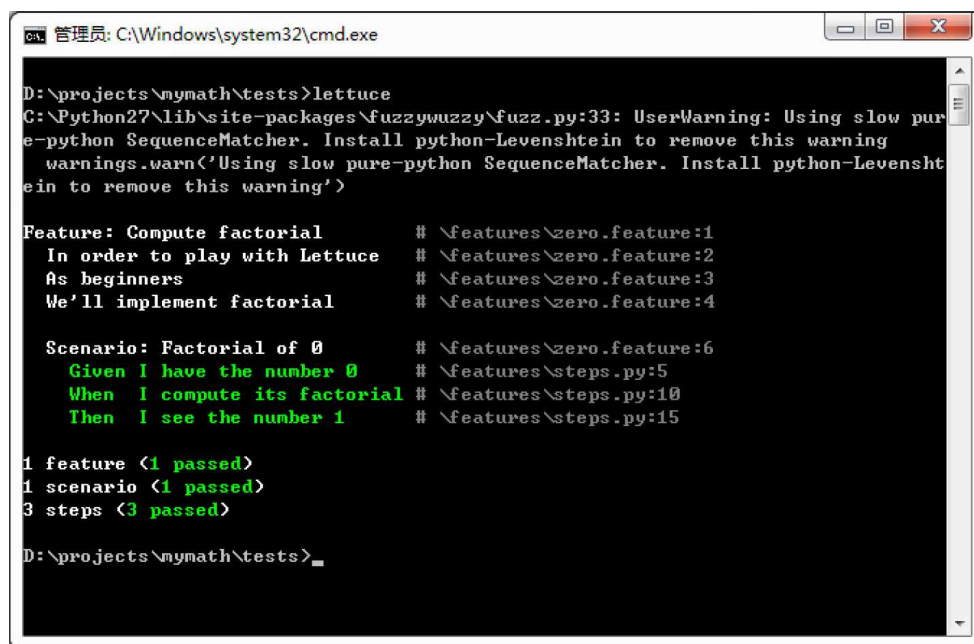
```
@step('I see the number (\d+)')
def check_number(step, expected):
    expected = int(expected)
    assert world.number == expected, "Got %d" % world.number
```

expected zero.feature  
world.number assert

I see the number (\d+) zero.feature “Then I see the number 1”

## Lettuce

cmd tests “Lettuce” 12.3



```
管理员: C:\Windows\system32\cmd.exe

D:\projects\mymath\tests>lettuce
C:\Python27\lib\site-packages\fuzzywuzzy\fuzz.py:33: UserWarning: Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning')

Feature: Compute factorial      # \features\zero.feature:1
  In order to play with Lettuce # \features\zero.feature:2
  As beginners                  # \features\zero.feature:3
  We'll implement factorial     # \features\zero.feature:4

  Scenario: Factorial of 0      # \features\zero.feature:6
    Given I have the number 0   # \features\steps.py:5
    When I compute its factorial # \features\steps.py:10
    Then I see the number 1     # \features\steps.py:15

1 feature <1 passed>
1 scenario <1 passed>
3 steps <3 passed>

D:\projects\mymath\tests>
```

12.3 Lettuce

zero.feature feature  
scenario steps.py

Feature(1 passed)

Scenario(1 passed) 0 0000000

Steps(3 passed) 0 0000000

## 12.3.3 0000000

000000zero.feature0000000000000000

zero.feature

### Feature:

Compute factorial

In order to play with Lettuce

As beginners

We'll implement factorial

### Scenario:

Factorial of 0

#### Given

I have the number 0

#### When

I compute its factorial

#### Then

I see the number 1

**Scenario:**

Factorial of 1

**Given**

I have the number 1

**When**

I compute its factorial

**Then**

I see the number 1

**Scenario:**

Factorial of 2

**Given**

I have the number 2

**When**

I compute its factorial

**Then**



I see the number 2

**Scenario:**

Factorial of 3

**Given**

I have the number 3

**When**

I compute its factorial

**Then**

I see the number 6

“Lettuce”12.4

```
C:\Windows\system32\cmd.exe

Scenario: Factorial of 0      # \features\zero.feature:6
  Given I have the number 0  # \features\steps.py:4
  When I compute its factorial # \features\steps.py:8
  Then I see the number 1    # \features\steps.py:12

Scenario: Factorial of 1      # \features\zero.feature:11
  Given I have the number 1  # \features\steps.py:4
  When I compute its factorial # \features\steps.py:8
  Then I see the number 1    # \features\steps.py:12

Scenario: Factorial of 2      # \features\zero.feature:16
  Given I have the number 2  # \features\steps.py:4
  When I compute its factorial # \features\steps.py:8
  Then I see the number 2    # \features\steps.py:12

Scenario: Factorial of 3      # \features\zero.feature:21
  Given I have the number 3  # \features\steps.py:4
  When I compute its factorial # \features\steps.py:8
  Then I see the number 6    # \features\steps.py:12
Traceback (most recent call last):
  File "C:\Python27\lib\site-packages\lettuce\core.py", line 144, in __call__
    ret = self.function(self.step, *args, **kw)
  File "E:\test_project\tests\features\steps.py", line 14, in check_number
    assert world.number == expected, "Got %d" % world.number
AssertionError: Got 3

1 feature <0 passed>
4 scenarios <3 passed>
12 steps <1 failed, 11 passed>

List of failed scenarios:
  Scenario: Factorial of 3      # \features\zero.feature:21
```

## 12.4 Lettuce

3! 6

steps.py factorial() factorial()

steps.py

.....

```
def factorial(number):
    number = int(number)
    if (number == 0) or (number == 1):
        return 1
    else:
        return number*factorial(number-1)
```

011  
Lettuce12.5

```
C:\Windows\system32\cmd.exe

We'll implement factorial      # \features\zero.feature:4

Scenario: Factorial of 0      # \features\zero.feature:6
  Given I have the number 0   # \features\steps.py:4
  When I compute its factorial # \features\steps.py:8
  Then I see the number 1     # \features\steps.py:12

Scenario: Factorial of 1      # \features\zero.feature:11
  Given I have the number 1   # \features\steps.py:4
  When I compute its factorial # \features\steps.py:8
  Then I see the number 1     # \features\steps.py:12

Scenario: Factorial of 2      # \features\zero.feature:16
  Given I have the number 2   # \features\steps.py:4
  When I compute its factorial # \features\steps.py:8
  Then I see the number 2     # \features\steps.py:12

Scenario: Factorial of 3      # \features\zero.feature:21
  Given I have the number 3   # \features\steps.py:4
  When I compute its factorial # \features\steps.py:8
  Then I see the number 6     # \features\steps.py:12

1 feature <1 passed>
4 scenarios <4 passed>
12 steps <12 passed>
```

12.5 Lettuce

## 12.3.4 Lettuce

BDDFeatureStepFeaturefeatureGiven-When-ThenscenariosStepPythonFeatureGiven-When-ThenStepPython

Featurefeatures“could not find features at \features”Step

## 12.4 Lettuce\_webdriver

Lettuce\_webdriver Python Selenium WebDriver

### Lettuce

12.2

### Lettuce\_webdriver

Lettuce\_webdriver [https://pypi.python.org/pypi/Lettuce\\_webdriver](https://pypi.python.org/pypi/Lettuce_webdriver)

pip

cmd.exe

```
> pip install lettuce_webdriver
```

### nose

nose unittest Python Lettuce\_webdriver nose

nose <https://pypi.python.org/pypi/nose/>

nose pip

cmd.exe

```
> pip install nose
```

tests/features/

└─ step\_definitions/

| └─ setps.py

└─ support/

| └─ terrain.py

└─ baidu.feature

baidu.feature BDD

baidu.feature

## Feature:

Baidu search test case

## Scenario:

search selenium

## Given

I go to "http://www.baidu.com/"

**When**

I fill in field with id "kw" with "selenium"

**And**

I click id "su" with baidu once

**Then**

I should see "seleniumhq.org" within 2 second

**Then**

I close browser

#####step\_definitions#####

steps.py

```
# coding=utf-8
from lettuce import *
from lettuce_webdriver.util import assert_false
from lettuce_webdriver.util import AssertContextManager
```

```
def input_frame(browser, attribute):
    xpath = "//input[@id='%s']" % attribute
    elems = browser.find_elements_by_xpath(xpath)
    return elems[0] if elems else False
```

```
def click_button(browser, attribute):
    xpath = "//input[@id='%s']" % attribute
    elems = browser.find_elements_by_xpath(xpath)
    return elems[0] if elems else False
```

```
# 百度搜索
```

```
@step('I fill in field with id "(.*?)" with "(.*?)"')
```

```
def baidu_text(step, field_name, value):
    with AssertContextManager(step):
        text_field = input_frame(world.browser, field_name)
        text_field.clear()
        text_field.send_keys(value)
```

```
# 点击“百度”
```

```
@step('I click id "(.*?)" with baidu once')
```

```
def baidu_click(step, field_name):
    with AssertContextManager(step):
        click_field = click_button(world.browser, field_name)
        click_field.click()
```

```
# 关闭
```

```
@step('I close browser')
```

```

def close_browser(step):
    world.browser.quit()

support[...][terrain.py][...]

terrain.py

from selenium import webdriver
from lettuce import before, world
import lettuce_webdriver.webdriver

@before.all
def setup_browser():
    world.browser = webdriver.Firefox()

terrain[...][...]

[...]/tests/[...]“Lettuce”[...][...]12.6[...]
```



```
C:\Windows\system32\cmd.exe
D:\pyle\tests>lettuce

Feature: Go to baidu                                     # \features\baidu.feature
:1

    Scenario: search selenium                             # \features\baidu.feature
    :3
        Given I go to "http://www.baidu.com/"           # C:\Python27\lib\site-pa
        Given I go to "http://www.baidu.com/"           # C:\Python27\lib\site-pa
        packages\lettuce_webdriver\webdriver.py:76
        When I fill in field with id "kw" with "selenium" # \features\step_definiti
        When I fill in field with id "kw" with "selenium" # \features\step_definiti
        ons\setps.py:28
        And I click id "su" with baidu once               # \features\step_definiti
        And I click id "su" with baidu once               # \features\step_definiti
        ons\setps.py:36
        Then I should see "seleniumhq.org" within 2 second # C:\Python27\lib\site-pa
        Then I should see "seleniumhq.org" within 2 second # C:\Python27\lib\site-pa
        packages\lettuce_webdriver\webdriver.py:152
        I close browser                                   # \features\step_definiti
        I close browser                                   # \features\step_definiti
        ons\setps.py:44

1 feature <1 passed>
1 scenario <1 passed>
5 steps <5 passed>
```

## 12.6 Lettuce\_webdriver

biadu.feature

baidu.feature

**Feature:**

Baidu search test case

**Scenario:**

search selenium

**Given**

I go to "http://www.baidu.com/"

**When**

I fill in field with id "kw" with "selenium"

**And**

I click id "su" with baidu once

**Then**

I should see "seleniumhq.org" within 2 second

### **Scenario:**

search lettuce\_webdriver

**Given**

I go to "http://www.baidu.com/"

**When**

I fill in field with id "kw" with "webdriver"

**And**

I click id "su" with baidu once

**Then**

I should see "www.w3.org/TR/webdriver/" within 2 second

**Then**

I close browser

“Then I close browser”  
“Lettuce”12.7

```
C:\Windows\system32\cmd.exe

Then I should see "seleniumhq.org" within 2 second # C:\Python27\l
Then I should see "seleniumhq.org" within 2 second # C:\Python27\l
ib\site-packages\lettuce_webdriver\webdriver.py:152

Scenario: search selenium # \features\bai
du.feature:3
Given I go to "http://www.baidu.com/" # C:\Python27\l
Given I go to "http://www.baidu.com/" # C:\Python27\l
ib\site-packages\lettuce_webdriver\webdriver.py:76
When I fill in field with id "kw" with "webdriver" # \features\ste
When I fill in field with id "kw" with "webdriver" # \features\ste
p_definitions\setps.py:28
And I click id "su" with baidu once # \features\ste
And I click id "su" with baidu once # \features\ste
p_definitions\setps.py:36
Then I should see "www.w3.org/TR/webdriver/" within 2 second # C:\Python27\l
Then I should see "www.w3.org/TR/webdriver/" within 2 second # C:\Python27\l
ib\site-packages\lettuce_webdriver\webdriver.py:152
I close browser # \features\ste
I close browser # \features\ste
p_definitions\setps.py:44

1 feature <1 passed>
2 scenarios <2 passed>
9 steps <9 passed>

D:\pyle\tests>
```

12.7 Lettuce\_webdriver

BDDLettuceBDD  
Lettuce\_webdriverWeb  
BDD